7

Microprocessoren

Inhoud

■/■■ reeds gepubliceerd
▒/▒▒ gepland voor de volgende aanvullingen

7/2

Acht bits processoren

Inhoud

7/2.1

6502/6510

De 6502 is een veelgebruikte processor, voornamelijk in home-computers zoals de Apple, BBC en Commodore. Deze processor is ontwikkeld door MOS Technology, een bedrijf dat later door Commodore is overgenomen. Het blokschema van de 6502 is in figuur 7/2.1-1 te zien.

Kenmerken van de 6502 zijn:

- 64 kB adresgebied;
- geïntegreerde klokgenerator, maximaal 2 MHz;
- twee 8-bits indexregisters;
- 8-bits stapelregister (stack);
- een maskeerbare en een niet maskeerbare interrupt;
- één voedingsspanning (5 V);
- zero-page addressing.

De 6502 instructieset

Opdracht	Object •	Byte	Cycli	Status						Uitgevoerde bewerkingen	
				N	V	D	I	Z	C		
LDA										Sla accumulator in geheugen op	
addr	A5 pp	2	3	X				X		A←[addr]	Pagina nul direkt
addr,X	B5 pp	2	4	X				X		A←[addr+X]	Pagina nul geïndiceerd
(addr,X)	A1 pp	2	6	X				X		A←[[addr+X]]	Voor-geïndiceerd indirekt
(addr),Y	B1 pp	2	5*	X				X		A←[[addr+1,addr]+Y]	Na-geïndiceerd indirekt
addr16	AD ppqq	3	4	X				X		A←[addr16]	Uitgebreid direkt
addr16,X of Y	11011x01 ppqq	3	4*	X				X		A←[addr16+X] of A←[addr16+Y]	Absoluut geïndiceerd
STA										Laad accumulator in geheugen	
addr	85 pp	2	3							[addr]←A	Pagina nul direkt
addr,X	95 pp	2	4							[addr+X]←A	Pagina nul geïndiceerd
(addr,X)	81 pp	2	6							[[addr+X]]←A	Voor-geïndiceerd indirekt
(addr),Y	91 pp	2	6							[[addr+1,addr]+Y]←A	Na-geïndiceerd indirekt
add16	8D ppqq	3	4							[addr16]←A	Uitgebreid direkt
addr16,X of Y	10011x01 ppqq	3	5							[addr16+X] of A←[addr16+Y]←A	Absoluut geïndiceerd
LDX										Laad index-register X vanuit geheugen,	index alleen over register Y
addr	A6 pp	2	3	X				X		X←[addr]	Pagina nul direkt
addr,Y	B6 pp	2	4	X				X		X←[addr+Y]	Pagina nul geïndiceerd
addr16	AE ppqq	3	4	X				X		X←[addr16]	Uitgebreid geïndiceerd
addr16,Y	BE ppqq	3	4*	X				X		X←[addr16+Y]	Absoluut geïndiceerd
STX										Sla index-register X *in het geheugen op,*	*index alleen over register Y*
addr	86 pp	2	3							[addr]←X	Pagina nul direkt
addr,Y	96 pp	2	4							[addr+Y]←X	Pagina nul geïndiceerd
addr16	8E ppqq	3	4							[addr16]←X	Uitgebreid direkt
LDY										Laad index-register Y vanuit geheugen,	index alleen over register X
addr	A4 pp	2	3	X				X		Y←[addr]	Pagina nul direkt
addr,X	B4 pp	2	4	X				X		Y← [addr+X]	Pagina nul geïndiceerd
addr16	AC ppqq	3	4	X				X		Y←[addr16]	Uitgebreid direkt
addr16,X	BC ppqq	3	4*	X				X		Y←[addr16+X]	Absoluut geïndiceerd
STY										Sla index-register Y in het geheugen op,	index alleen over register X
addr	84 pp	2	3							[addr]←Y	Pagina nul direkt
addr,X	94 pp	2	4							[addr+X]←Y	Pagina nul geïndiceerd
addr16	8C ppqq	3	4							[addr16]←Y	Uitgebreid direkt

Tabel 7/2.1-1: De instructie-set van de 6502.

2.1 6502/6510

Opdracht	Object •	Byte	Cycli	Status N	V	D	I	Z	C	Uitgevoerde bewerkingen	
ADC										Tel inhoud van geheugenplaats met overdracht op bij die van accumul.	
addr	65 pp	2	3	X	X			X	X	A←A+[addr]+C	Pagina nul direkt
addr,X	75 pp	2	4	X	X			X	X	A←A+[addr+X]+C	Pagina nul geïndiceerd
(addr,X)	61 pp	2	6	X	X			X	X	A←A+[[addr+X]]+C	Voor-geïndiceerd indirekt
(addr),Y	71 pp	2	5*	X	X			X	X	A←A+[[addr+1,addr]+Y]+C	Na-geïndiceerd indirekt
add16	6D ppqq	3	4	X	X			X	X	A←A+[addr16]+C	Uitgebreid direkt
addr16,X of Y	01111x01 ppqq	3	4*							A←A+[addr16+X]+C of A←A+[addr16+Y]+C	Absoluut geïndiceerd
										(Nul-vlag is in de decimale mode niet geldig)	
AND										AND de inhoud van de accumulator met die van een geheugenplaats.	
addr	25 pp	2	3	X				X		A←A∧[addr]	Pagina nul direkt
addr,X	35 pp	2	4	X				X		A←A∧[addr+Y]	Pagina nul geïndiceerd
(addr,X)	21 pp	2	6	X				X		A←A∧[[addr+X]]	Voor-geïndiceerd indirekt
(addr),Y	31 pp	2	5*	X				X		A←A∧[[addr+1, addr]+Y	Na-geïndiceerd indirekt
addr16	2D ppqq	3	4	X				X		A←A∧[addr16]	Uitgebreid direkt
addr16,X of Y	00111x01 ppqq	3	4*	X				X		A←A+[addr16+X]+C of A←A+[addr16+Y]	Absoluut geïndiceerd
BIT										AND de accumulator-inhoud met die van een geheugenplaats. Alleen de statusbits worden hierbij beïnvloed.	
addr	24 pp	2	3	7	6			X		A∧[addr]	Pagina nul direkt
addr16	2C ppqq	3	4	7	6			X		A∧[addr16]	Uitgebreid direkt
CMP										Vergelijk de accumulator-inhoud met die in de geheugenplaats. Alleen de statusbits worden beïnvloed.	
addr	C5 pp	2	3	X				X	X	A–[addr]	Pagina nul direkt
addr,X	D5 pp	2	4	X				X	X	A–[addr+X]	Pagina nul geïndiceerd
(addr,X)	C1 pp	2	6	X				X	X	A–[[addr+X]]	Voor-geïndiceerd indirekt
(addr),Y	D1 pp	2	5*	X				X	X	A–[[addr+1, addr]+Y]	Na-geïndiceerd indirekt
addr16	CD ppqq	3	4	X				X	X	A–[addr16]	Uitgebreid direkt
addr16,X of Y	11011x01 ppqq	3	4*	X				X	X	A←A+[addr16+X]+C of A←A+[addr16+Y]	Absoluut geïndiceerd
EOR										XOR de accumulator-inhoud met die van de geheugenplaats.	
addr	45 pp	2	3	X				X		A←A⊻[addr]	Pagina nul direkt
addr,X	55 pp	2	4	X				X		A←A⊻[addr+X]	Pagina nul geïndiceerd
(addr,X)	41 pp	2	6	X				X		A←A⊻[[addr+X]]	Voor-geïndiceerd indirekt
(addr),Y	51 pp	2	5*	X				X		A←A⊻[[addr+1, addr]+Y]	Na-geïndiceerd indirekt
addr16	4D ppqq	3	4	X				X		A←A⊻[addr16]	Uitgebreid direkt
addr16,X of Y	01011x01 ppqq	3	4*	X				X		A←A+[addr16+X]+C of A←A+[addr16+Y]	Absoluut geïndiceerd
ORA										OR de accumulator-inhoud met die van de geheugenplaats	
addr	05 pp	2	3	X				X		A←AV[addr]	Pagina nul direkt
addr,X	15 pp	2	4	X				X		A←AV[addr+X]	Pagina nul geïndiceerd
(addr,X)	01 pp	2	6	X				X		A←AV([addr+X]]	Voor-geïndiceerd indirekt
(addr),Y	11 pp	2	5*	X				X		A←AV[addr+1, addr]+Y]	Na-geïndiceerd indirekt
addr16	0D ppqq	3	4	X				X		A←AV[addr16]	Uitgebreid direkt
addr16,X of Y	00011x01 ppqq	3	4*	X				X		A←A+[addr16+X]+C of A←A+[addr16+Y]	Absoluut geïndiceerd
SBC										Verminder inhoud geheugenplaats - met lenen - met die van accumulator	
addr	E5 pp	2	3	X	X			X	X	A←A–[addr]–C̄	Pagina nul direkt
addr,X	F5 pp	2	4	X	X			X	X	A←A–[addr+X]–C̄	Pagina nul geïndiceerd
(addr,X)	E1 pp	2	6	X	X			X	X	A←A–[[addr+X]]–C̄	Voor-geïndiceerd indirekt
(addr,)Y	F1 pp	2	5*	X	X			X	X	A←A–[[addr+1, addr]+Y]–C̄	Na-geïndiceerd indirekt
addr16	ED ppqq	3	4	X	X			X	X	A←A–[addr16]–C	Uitgebreid direkt
addr16,X of Y	11111x01 ppqq	3	4*	X	X			X	X	A←A+[addr16+X]+C̄ of A←A+[addr16+Y]+C̄	Absoluut geïndiceerd
										(Let erop, dat de overdrachtwaarde het komplement is van het leenbedrag is).	
INC										Incrementeer inhoud geheugenplaats. Index alleen via register X.	
addr	E6 pp	2	5	X				X		[addr]←[addr]+1	Pagina nul direkt
addr,X	F6 pp	2	6	X				X		[addr+X]←[addr+Y]+1	Pagina nul geïndiceerd
addr16	EE ppqq	3	6	X				X		[addr16]←[addr16]+1	Uitgebreid direkt
addr16,X	FE ppqq	3	7	X				X		[addr16+X]←[addr16+X]+1	Absoluut geïndiceerd
DEC										Decrementeer inhoud geheugenplaats. Index alleen via register X.	
addr	C6 pp	2	5	X				X		[addr]←[addr]-1	Pagina nul direkt
addr,X	D6 pp	2	6	X				X		[addr+X]←[addr+X]-1	Pagina nul geïndiceerd
addr16	CE ppqq	3	6	X				X		[addr16]←[addr16]-1	Uitgebreid direkt
addr16,X	DE ppqq	3	7	X				X		[addr16+X]-1	Absoluut geïndiceerd
CPX										Vergelijk inhoud X-register met die van de geheugenplaats. Alleen de statusvlaggen worden beïnvloed.	
addr	E4 pp	2	3	X				X	X	X–[addr]	Pagina nul direkt
addr16	EC ppqq	3	4	X				X	X	X–[addr16]	Uitgebreid direkt
CPY										Vergelijk inhoud Y-register met die van de geheugenplaats. Alleen de statusvlaggen worden beïnvloed.	
addr	C4 pp	2	3	X				X	X	Y–[addr]	Pagina nul direkt
addr16	CC ppqq	3	4	X				X	X	Y–[addr16]	Uitgebreid direkt

Figuur 7/2.1-2: Interne opbouw van de 6502 processor.

2.1 6502/6510

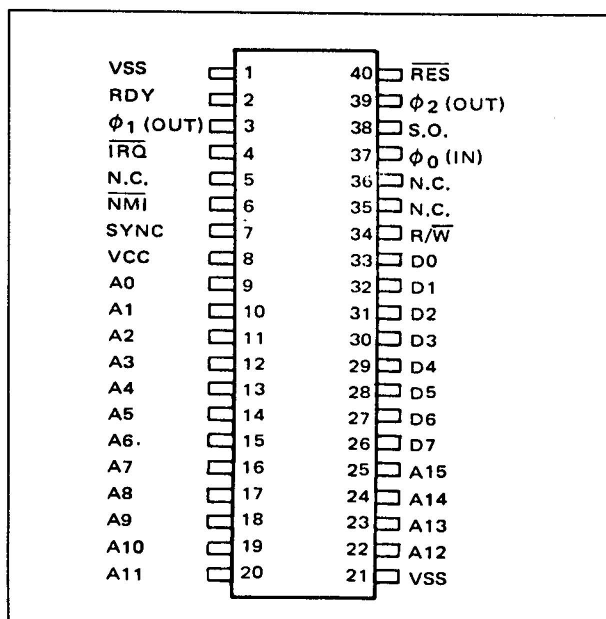
Opdracht	Object •	Byte	Cycli	Status N	Status V	Status D	Status I	Status Z	Status C	Uitgevoerde bewerkingen
ROL										Roteer inhoud geheugenplaats een bit naar links door de overdracht. Index alleen door register X.
addr	26 pp	2	5	X				X	X	|addr|
addr,X	36 pp	2	6	X				X	X	|addr+X|
addr16	2E ppqq	3	6	X				X	X	|addr16|
addr16,X	3E ppqq	3	7	X				X	X	|addr16+X|
ROR										Roteer inhoud geheugenplaats een bit naar rechts. Index alleen door register X.
addr	66 pp	2	5	X				X	X	|addr|
addr,X	76 pp	2	6	X				X	X	|addr+X|
addr16	6E pp	3	6	X				X	X	|addr16|
addr16,X	7E ppqq	3	7	X				X	X	|addr16+X|
ASL										Schuif inhoud geheugenplaats arithmetisch naar links. Index alleen door register X.
addr	06 pp	2	5	X				X	X	|addr|
addr,X	16 pp	2	6	X				X	X	|addr+X|
addr16	0E ppqq	3	6	X				X	X	|addr16|
addr16,X	1E ppqq	3	7	X				X	X	|addr16+X|
LSR										Schuif inhoud geheugenplaats logisch naar rechts. Index alleen door register X.
addr	46 pp	2	5	0				X	X	|addr|
addr,X	56 pp	2	6	0				X	X	|addr+X|
addr16	4E ppqq	3	6	0				X	X	|addr16|
addr16,X	5E ppqq	3	7	0				X	X	|addr16.X|
LDA data	A9 pp	2	2	X				X		Laad accumulator met direkte gegevens. A←data
LDX data	A2 pp	2	2	X				X		Laad index-register X met direkte gegevens. X←data
LDY data	A0 pp	2	2	X				X		Laad index-register Y met direkte gegevens. Y←data
ADC data	69 pp	2	2	X	X			X	X	Tel direkt, met overdracht, op met accumulator. De nul-vlag is bij decimaalbedrijf niet geldig. A←A+data+C
AND data	29 pp	2	2	X				X		AND direkt met accumulator. A←A data
CMP data	C9 pp	2	2	X				X	X	Vergelijk direkt met accumulator. Alleen de statusvlaggen worden beïnvloed. A-data
EOR data	49 pp	2	2	X				X		XOR direkt met accumulator. A←A ⊻ data
ORA data	09 pp	2	2	X				X		OR direkt met accumulator. A←A V data
SBC data	E9 pp	2	2	X	X			X	X	Verminder direkt, met lenen, van accumulator. A←A-data-C
CPX data	E0 pp	2	2	X				X	X	(Let erop, dat de overdrachtwaarde het komplement is van het leenbedrag). Vergelijk direkt met index-register X. Alleen de statusvlaggen worden beïnvloed. X-data
CPY data	C0 pp	2	2	X				X	X	Vergelijk direkt met index-register Y. Alleen de statusvlaggen worden beïnvloed. Y-data
JMP label	4C ppqq	3	3							Spring naar volgende geheugenplaats, middels uitgebreide of indirekte adressering. PC←label of PC←(label)
(label)	6C ppqq	3	5							Let op het volgende voor alle voorwaardelijke vertakkingsopdrachten: wordt aan de voorwaarde voldaan, dan wordt de verplaatsing in de opdrachtenteller opgeteld, nadat de opdrachtenteller geïndiceerd is, om de opdracht te kunnen laten zien die op de vertakkingsopdracht volgt

2.1 6502/6510

Opdracht	Object •	Byte	Cycli	Status N	V	D	I	Z	C	Uitgevoerde bewerkingen
BCC disp	90 pp	2	2**							Vertak relatief, nadat overdrachtsvlag gewist is. Indien C-0, dan PC–PC+disp
BCS disp	B0 pp	2	2**							Vertak relatief, nadat overdrachtsvlag gezet is. Indien C-1, dan PC–PC+disp
BEQ disp	F0 pp	2	2**							Vertak relatief, wanneer resultaat gelijk is aan nul. Indien Z-1, dan PC–PC+disp
BMI disp	30 pp	2	2**							Vertak relatief, wanneer resultaat negatief is. Indien N-1, dan PC–PC+disp
BNE disp	D0 pp	2	2**							Vertak relatief, wanneer resultaat ongelijk nul is. Indien Z-0, dan PC–PC+disp
BPL disp	10 pp	2	2**							Vertak relatief, wanneer resultaat positief is. Indien N-0, dan PC–PC+disp
BVC disp	50 pp	2	2**							Vertak relatief, wanneer overloopvlag gewist is. Indien V-0, dan PC–PC+disp
BVS disp	70 pp	2	2**							Vertak relatief, wanneer overloopvlag gezet is. Indien V-1, dan PC–PC+disp
JSR label	20 ppqq	3	6							Spring naar subroutine, te beginnen bij adres dat door de bytes 2 en 3 van de instructie aangegeven is. Let erop, dat de opgeslagen waarde voor de opdrachtenteller in zijn geheel de JSR-opdracht bevat. [SP]←PC(HI) [SP-1]←PC(LO) SP←SP-2 PC←label
RTS	60	1	6							Keer van subroutine terug en incrementeer hierbij de opdrachtenteller, om die naar de opdracht na JSR te laten wijzen die de subroutine tot gevolg had. PC(LO)←[SP+1] PC(HI)←[SP+2] SP←SP+2 PC←PC+1
TAX	AA	1	2	x				x		Breng de accumulator-inhoud over naar indexregister X. X←A
TXA	8A	1	2	x				x		Breng inhoud indexregister X over naar accumulator. A←X
TAY	A8	1	2	x				x		Breng inhoud accumulator over naar indexregister Y. Y←A
TYA	98	1	2	x				x		Breng inhoud indexregister Y over naar accumulator. A←Y
TSX	BA	1	2	x				x		Breng inhoud stapelaanwijzer over naar indexregister X. X←SP
TXS	9A	1	2							Breng inhoud indexregister X over naar stapelaanwijzer. SP←X
DEX	CA	1	2	x				x		Decrementeer inhoud indexregister X. X←X-1
DEY	88	1	2	x				x		Decrementeer inhoud indexregister Y. Y←Y-1
INX	E8	1	2	x				x		Incrementeer inhoud indexregister X. X←X+1
INY	C8	1	2	x				x		Incrementeer inhoud indexregister Y. Y←Y+1
ROL A	2A	1	2	x				x	x	Roteer inhoud accumulator naar links, door overdracht.
ROR A	6A	1	2	x				x	x	Roteer inhoud accumulator naar rechts, door overdracht.
ASL A	0A	1	2	x				x	x	Schuif inhoud accumulator arithmetisch naar links.
LSR A	4A	1	2	0				x	x	Schuif inhoud accumumator arithmetisch naar rechts.
PHA	48	1	3							Breng inhoud accumulator over naar stapel. [SP]←A SP←SP-1
PLA	68	1	4	x				x		Laad accumulator vanaf de top van de stapel ("Pull"). A←[SP+1] SP←SP+1

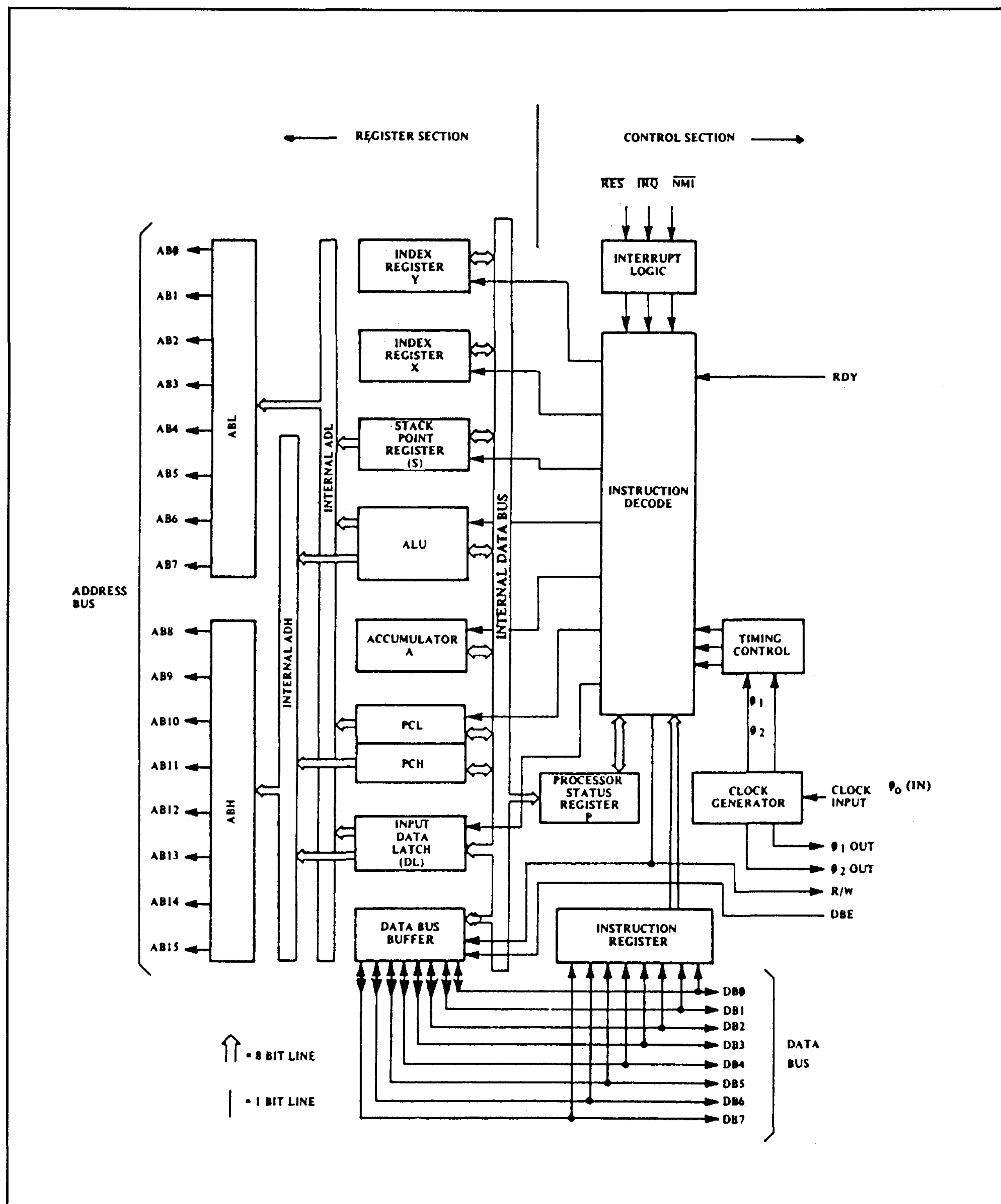
2.1 6502/6510

Opdracht	Object •	Byte	Cycli	Status N	V	D	I	Z	C	Uitgevoerde bewerkingen
PHP	08	1	3							Breng inhoud statusregister over naar stapel. [SP]←P SP←SP-1
PLP	28	1	4	X	X	X	X	X	X	Laad statusregister vanaf top van stapel ("Pull"). P←[SP+1] SP←SP+1
CLI	58	1	2				0			Maak onderbrekingen, door het terugstellen van de betreffende interrupt-bit in het statusregister, vrij. I←0
SEI	78	1	2				1			Maak onderbrekingen onmogelijk. I←1
RTI	40	1	6	X	X	X	X	X	X	Keer van onderbreking terug, herstel status. P←[SP+1] PC(LO)←[SP+2] PC(HI)←[SP+3] SP←SP+3 PC←PC+1
BRK	00	1	7				1			Geprogrammeerde onderbrekingen. BRK is niet te sperren. De programmateller wordt twee maal geïncrementeerd, voordat deze in de stapel opgeslagen wordt. [SP]←PC(HI) [SP-1]←PC(LO) [SP-2]←P SP←SP-3 PC(HI)←[FFFF] PC(LO)←[FFFE] I←1 B←2
CLC	18	1	2						0	Stel overdrachtsvlag terug. C←0
SEC	38	1	2						1	Zet overdrachtsvlag. C←1
CLD	D8	1	2			0				Stel decimaal-mode terug. D←0
SED	F8	1	2			1				Zet decimaal-mode D←1
CLV	B8	1	2		0					Stel overloopvlag terug. V←0
NOP	EA	1	2							Geen bewerking.

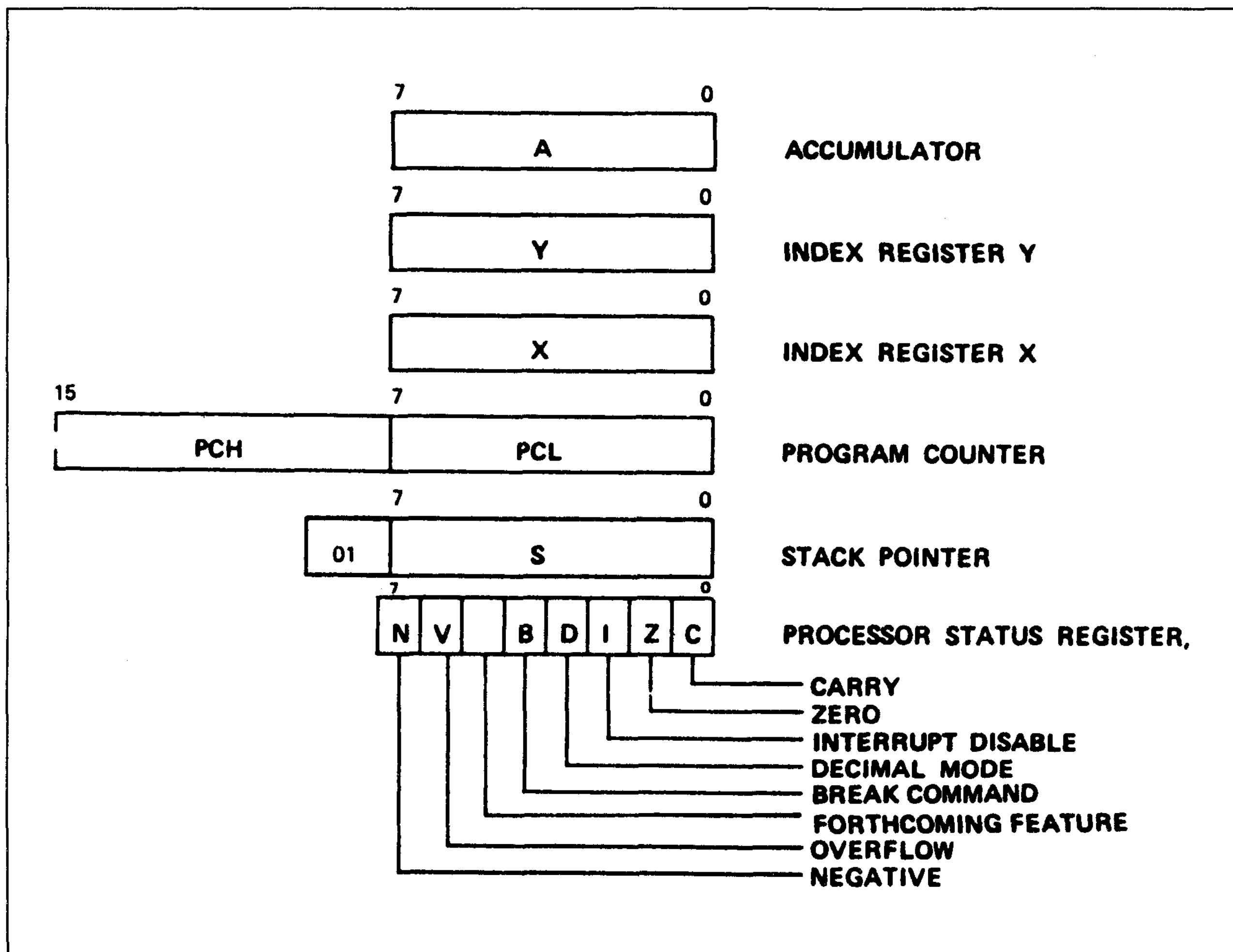


Figuur 7/2.1-1: Aansluitgegevens van de 6502.

2.1 6502/6510



Figuur 7/2.1-2: Interne opbouw van de 6502 processor.



Figuur 7/2.1-3 De register van de 6502.

Een extra kenmerk van de 6510, welke verder identiek aan de 6502 is, is:

- zes I/O-lijnen op de chip aanwezig, adresseerbaar via geheugenplaats 0000 (richtingsregister) en geheugenplaats 0001 (dataregister).

Addressing modes

De 6502 kent diverse addressing modes, waaronder de snelle zero-page addressing mode. Een kort overzicht volgt hier:

- zero-page addressing
  In deze mode bestaat de operand uit één byte welke een adres in de zero-page (bladzijde 0, geheugenplaatsen 0 t/m 255, d.w.z. $00 t/m $FF) aanwijst. Hier staat de te bewerken data. Schrijfwijze: INS addr (INS=instructie)
- indexed zero-page addressing
  Zie boven echter het X-register wordt eerst bij de operand opgeteld. Schrijfwijze: INS addr, X
- indirect indexed addressing
  De operand opgeteld bij het X-register wijst een adres aan dat het lowbyte van het doeladres bevat. Eén plaats verder staat het highbyte. Deze bytes samen vormen het doeladres voor de te bewerken data. Schrijfwijze: INS (addr,X)
- indexed indirect addressing
  De operand wijst een adres aan waar het lowbyte van het doeladres staat. Een plaats verder staat het highbyte. Bij het aldus

2.1 6502/6510

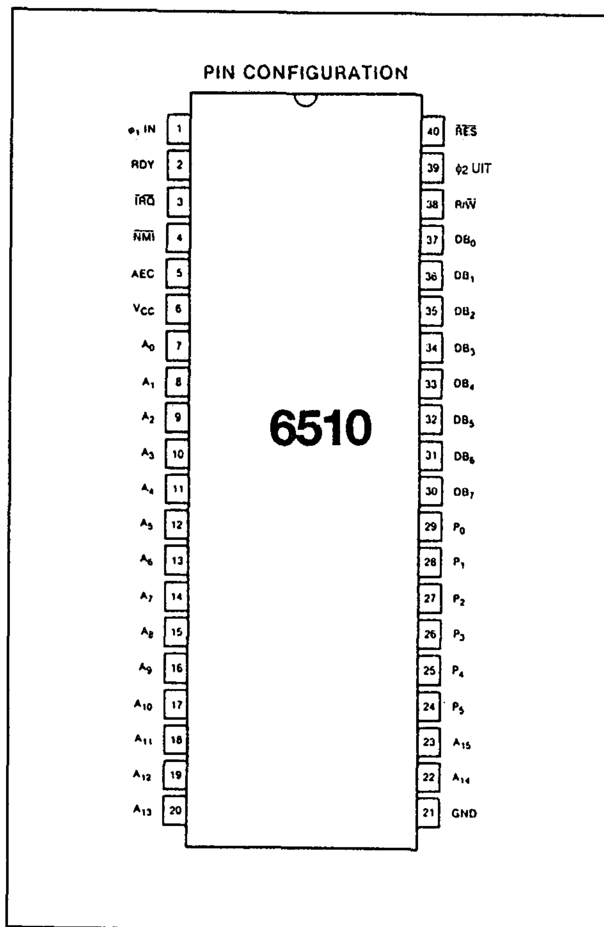
gevonden doeladres wordt het Y-register opgeteld. De aldus gevonden waarde bepaalt de locatie van de te bewerken data. Schrijfwijze: INS (addr),Y

- absolute addressing
  De operand staat in een geheugenlocatie die door de twee volgende bytes bepaald wordt. Eerst volgt het low-byte, daarna het highbyte. Schrijfwijze: INS addr16
- indexed absolute addressing
  Als boven, echter het het X- of Y-register wordt eerst bij het gevonden adres opgeteld. Schrijfwijze: INS addr16,X of INS addr16,Y
- Immediate addressing
  De operand staat in de byte na de instructie. Schrijfwijze: INS data
- Implied addressing
  De operand bevindt zich in de instructie zelf, er is dus geen aparte operand. Deze instructie zijn altijd slechts 1 byte lang. Schrijfwijze: INS
- Accumulator addressing
  De operand is de accu, deze wordt door de letter A aangeduid. De instructie is echter 1 byte lang, het is dus een vorm van implied addressing (zie boven). Schrijfwijze: INS A
- Relative addressing
  Deze addressing mode wordt alleen door de branch-instructies gebruikt (BCC, BEQ etcetera). De operand wijst een adres aan waar naartoe gesprongen (gebrancht) wordt, dat wil zeggen waar het programma verder gaat. Is de operand kleiner dan 128 dan wordt er het aangeduide aantal plaatsen vooruit gesprongen, is de operand groter of gelijk aan 128 dan wordt deze eerste van 255 afgetrokken alvorens er het aldus gevonden aantal plaatsen teruggesprongen wordt. Het tellen van het aantal te springen plaatsen begint altijd bij de instructie die op de branch-instructie volgt.

Registers

De 6502 heeft 6 registers. Dit zijn de accu (A), de beide indexregisters (X en Y), de stackpointer, de program counter (PC) en het statusregister (PS). Alle registers zijn 8 bits breed, behalve de PC, deze is 16 bits breed. De (virtuele) 8 hoogste bits van de stackpointer zijn gelijk aan 00000001, dit betekent dat de stack altijd van \$0100 t/m \$01FF loopt en dus maximaal 256 bytes groot is. De voorafgaande figuur geeft de indeling van de registers:

2.1 6502/6510/65C02



Figuur 7/2.1-4: Aansluitgegevens van de 6510.

65C02

Van de veelgebruikte 6502 microprocessor worden nu door Rockwell en GTE ook CMOS-uitvoeringen vervaardigd, die een hogere snelheid combineren met een geringer opgenomen vermogen. Bovendien werd de instructieset uitgebreid en zijn enkele nieuwe adresseringsmodes mogelijk.

LET OP
De R65C02 van Rockwell heeft 12 nieuwe instructies en de G65SC02 van GTE heeft er 8! Beide processoren kennen echter wel de instructies van de 'oude' versie.

Kenmerken*)
- alle eigenschappen van de 6502, plus
- CMOS silicon gate technologie
- geringe dissipatie: 4 mA/MHz
- 'neerwaarts software compatibel' met de 6502
  12 (8) nieuwe instructies
  2 nieuwe adresseringsmodes
- enkele +5 V voeding ± 20% (±10%)
- klokfrequentie (1), 2, 3 of 4 MHz

*) tussen haakjes de voor de G65SC02 geldende waarden.

De nieuwe instructies
- BBR Branch on Bit Reset (F)
- BBS Branch on Bit Set (F)
- BRA Branch Always
- PHX Push X Register on Stack
- PHY Push Y Register on Stack
- PLX Pull X Register from Stack
- PLY Pull Y Register from Stack
- RMB Reset Memory Bit (7)
- SMB Set Memory Bit (7)
- STZ Store Zero
- TRB Test and Reset Bits
- TSB Test and Set Bits

De met (F) en (7) gemerkte opcodes zijn opgenomen in de kolommen F en 7 van de opcode-matrix (die bij de G65SC02 niet worden gebruikt, zodat deze opcodes hierbij dan ook ontbreken).

2.1 6502/6510/65C02

MSD \ LSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRK Implied 1 7	ORA (IND, X) 2 6			TSB ZP 2 5	ORA ZP 2 3	ASL ZP 2 5	RMB0 ZP 2 5	PHP Implied 1 3	ORA IMM 2 2	ASL Accum 1 2		TSB ABS 3 6	ORA ABS 3 4	ASL ABS 3 6	BBR0 ZP 3 5**	0
1	BPL Relative 2 2**	ORA (IND), Y 2 5*	ORA (IND) 2 5		TRB ZP 2 5	ORA ZP, X 2 4	ASL ZP, X 2 6	RMB1 ZP 2 5	CLC Implied 1 2	ORA ABS, Y 3 4*	INC Accum 1 2		TRB ABS 3 6	ORA ABS, X 3 4*	ASL ABS, X 3 7	BBR1 ZP 3 5**	1
2	JSR Absolute 3 6	AND (IND, X) 2 6			BIT ZP 2 3	AND ZP 2 3	ROL ZP 2 5	RMB2 ZP 2 5	PLP Implied 1 4	AND IMM 2 2	ROL Accum 1 2		BIT ABS 3 4	AND ABS 3 4	ROL ABS 3 6	BBR2 ZP 3 5**	2
3	BMI Relative 2 2**	AND (IND), Y 2 5*	AND (IND) 2 5		BIT ZP, X 2 4	AND ZP, X 2 4	ROL ZP, X 2 6	RMB3 ZP 2 5	SEC Implied 1 2	AND ABS, Y 3 4*	DEC Accum 1 2		BIT ABS, X 3 4*	AND ABS, X 3 4*	ROL ABS, X 3 7	BBR3 ZP 3 5**	3
4	RTI Implied 1 6	EOR (IND, X) 2 6				EOR ZP 2 3	LSR ZP 2 5	RMB4 ZP 2 5	PHA Implied 1 3	EOR IMM 2 2	LSR Accum 1 2		JMP ABS 3 3	EOR ABS 3 4	LSR ABS 3 6	BBR4 ZP 3 5**	4
5	BVC Relative 2 2**	EOR (IND), Y 2 5*	EOR (IND) 2 5			EOR ZP, X 2 4	LSR ZP, X 2 6	RMB5 ZP 2 5	CLI Implied 1 2	EOR ABS, Y 3 4*	PHY Implied 1 2			EOR ABS, X 3 4*	LSR ABS, X 3 7	BBR5 ZP 3 5**	5
6	RTS Implied 1 6	ADC (IND, X) 2 6†			STZ ZP 2 3	ADC ZP 2 3†	ROR ZP 2 5	RMB6 ZP 2 5	PLA Implied 1 4	ADC IMM 2 2†	ROR Accum 1 2		JMP Indirect 3 5	ADC ABS 3 4†	ROR ABS 3 6	BBR6 ZP 3 5**	6
7	BVS Relative 2 2**	ADC (IND), Y 2 5*†	ADC (IND) 2 5†		STZ ZP, X 2 4	ADC ZP, X 2 4†	ROR ZP, X 2 6	RMB7 ZP 2 5	SEI Implied 1 2	ADC ABS, Y 3 4*†	PLY Implied 1 2		JMP (IND), X 3 6	ADC ABS, X 3 4*†	ROR ABS, X 3 7	BBR7 ZP 3 5**	7
8	BRA Relative 2 3	STA (IND, X) 2 6			STY ZP 2 3	STA ZP 2 3	STX ZP 2 3	SMB0 ZP 2 5	DEY Implied 1 2	BIT IMM 2 2	TXA Implied 1 2		STY ABS 3 4	STA ABS 3 4	STX ABS 3 4	BBS0 ZP 3 5**	8
9	BCC Relative 2 2**	STA (IND), Y 2 6	STA (IND) 2 6		STY ZP, X 2 4	STA ZP, X 2 4	STX ZP, Y 2 4	SMB1 ZP 2 5	TYA Implied 1 2	STA ABS, Y 3 5	TXS Implied 1 2		STZ ABS 3 4	STA ABS, X 3 5	STZ ABS, X 3 5	BBS1 ZP 3 5**	9
A	LDY IMM 2 2	LDA (IND, X) 2 6	LDX IMM 2 2		LDY ZP 2 3	LDA ZP 2 3	LDX ZP 2 3	SMB2 ZP 2 5	TAY Implied 1 2	LDA IMM 2 2	TAX Implied 1 2		LDY ABS 3 4	LDA ABS 3 4	LDX ABS 3 4	BBS2 ZP 3 5**	A
B	BCS Relative 2 2**	LDA (IND), Y 2 5*	LDA (IND) 2 5		LDY ZP, X 2 4	LDA ZP, X 2 4	LDX ZP, Y 2 4	SMB3 ZP 2 5	CLV Implied 1 2	LDA ABS, Y 3 4*	TSX Implied 1 2		LDY ABS, X 3 4*	LDA ABS, X 3 4*	LDX ABS, Y 3 4*	BBS3 ZP 3 5**	B
C	CPY IMM 2 2	CMP (IND, X) 2 6			CPY ZP 2 3	CMP ZP 2 3	DEC ZP 2 5	SMB4 ZP 2 5	INY Implied 1 2	CMP IMM 2 2	DEX Implied 1 2		CPY ABS 3 4	CMP ABS 3 4	DEC ABS 3 6	BBS4 ZP 3 5**	C
D	BNE Relative 2 2**	CMP (IND), Y 2 5*	CMP (IND) 2 5			CMP ZP, X 2 4	DEC ZP, X 2 6	SMB5 ZP 2 5	CLD Implied 1 2	CMP ABS, Y 3 4*	PHX Implied 1 2			CMP ABS, X 3 4*	DEC ABS, X 3 7	BBS5 ZP 3 5**	D
E	CPX IMM 2 2	SBC (IND, X) 2 6†			CPX ZP 2 3	SBC ZP 2 3†	INC ZP 2 5	SMB6 ZP 2 5	INX Implied 1 2	SBC IMM 2 2†	NOP Implied 1 2		CPX ABS 3 4	SBC ABS 3 4†	INC ABS 3 6	BBS6 ZP 3 5**	E
F	BEQ Relative 2 2**	SBC (IND), Y 2 5*†	SBC (IND) 2 5†			SBC ZP, X 2 4†	INC ZP, X 2 6	SMB7 ZP 2 5	SED Implied 1 2	SBC ABS, Y 3 4*†	PLX Implied 1 2			SBC ABS, X 3 4*†	INC ABS, X 3 7	BBS7 ZP 3 5**	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

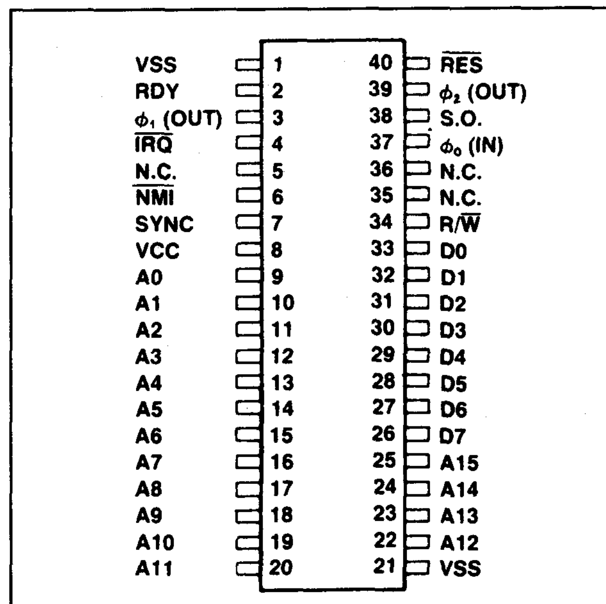
[ ] – Nieuwe opcode

Example cell (row 0, column 0): BRK / Implied / 1 7
- BRK – OP code
- Implied – adresserings mode
- 1 7 – instructie bytes; machine cyclussen

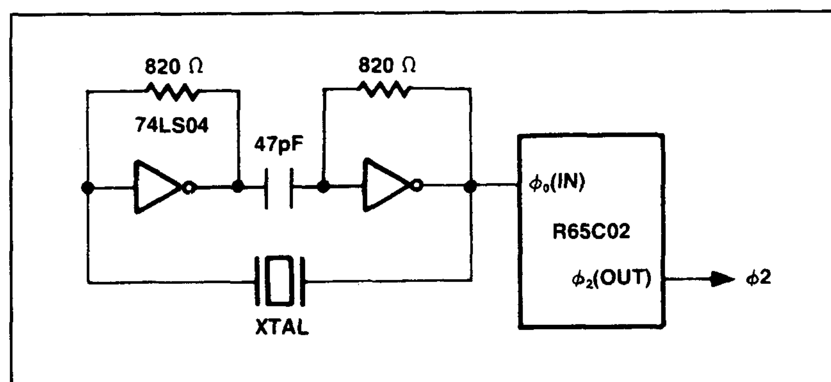
† tel hier 1 bij op indien in de decimale mode
* tel hier 1 bij op indien van pagina wordt gewisseld
** tel hier 1 bij op indien vertakking (branch) optreedt op dezelfde pagina (2 indien van pagina wordt gewisseld).

Tabel 7/2.1-2: Opcode-matrix van de 65C02 processor.

2.1 6502/6510/65C02



Figuur 7/2.1-5: Aansluitgegevens van de 65C02.



Figuur 7/2.1-6: Een veel gebruikte methode voor het opwekken van de systeemklok.

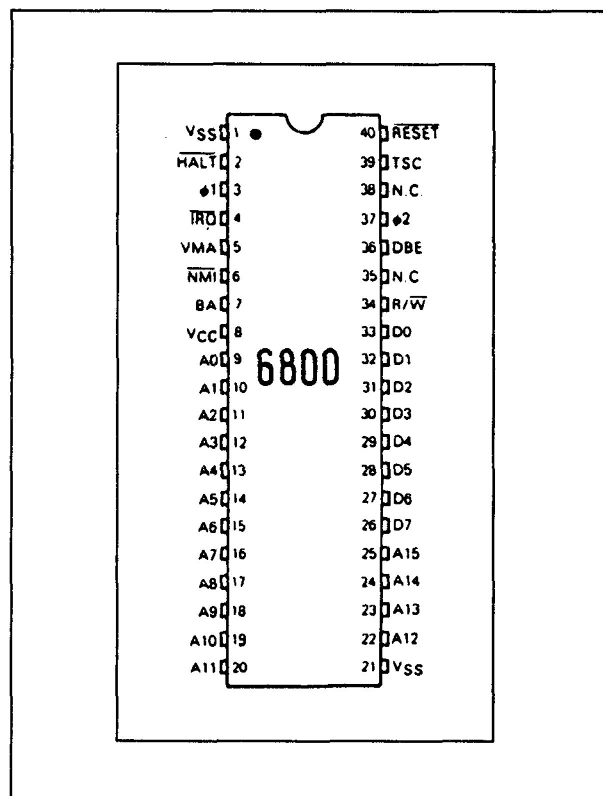
7/2.2

6800/6809

De MC 6800 werd in 1974 door Motorola op de markt gebracht en gold als eerste serieus te nemen concurrent voor de 8080 van Intel. Vanwege een andere opbouw heeft de 6800 geen afzonderlijke in- en uitvoer opdrachten. Alle periferie-modules worden binnen het normale adresseergebied aangestuurd (Memory-Mapped I/O). Andere kenmerken zijn:

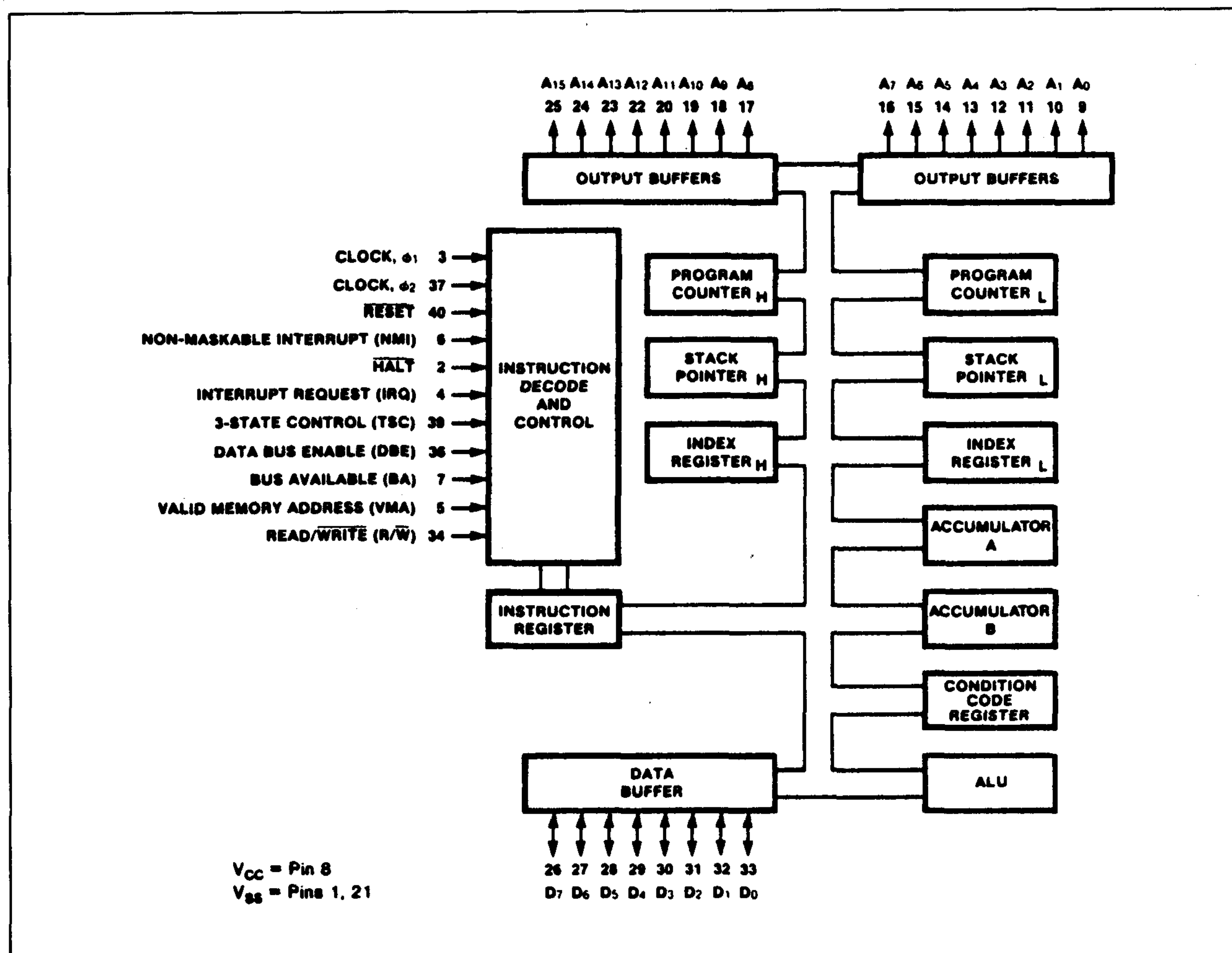
- overzichtelijke interne opbouw.
- stapelaanwijzer werkt met 16 bit (64K adresseergebied).
- indexregisters werken met 16 bit.
- twee parallelle accumulatoren, die tezamen 16 bit breed kunnen worden (mathematische en schuifopdrachten).
- relatieve berekening van de in te vullen adressen.
- klokfrequentie maximaal 2 MHz (B-versie).
- correcte adresbyte volorde (meest belangrijke, minst belangrijke byte).

De basis instructie-set omvat 72 bewerkingen, waaronder decimale arithmetic, 16 bit schuifbewerkingen en automatische bijhouden van de stack van registers bij interrupt- of subroutinesprongen.



Figuur 7/2.2-1: Aansluitgegevens van de 6800.

2.2 6800/6809



Figuur 7/2.2-2: Interne structuur van de 6800.

2.2 6800/6809

Machinecode	Words	Cycles ≡ States	Mnemonic	Verklaring	Schrijfwijze	Status-register
1X YY1 011	2/3	2/5	ADDX	Add memory to accumulator	(ACCX) + (M) → ACCX	H, N, Z, V, C
00 011 011	1	2	ABA	Add accumulator B to accumulator A	(ACCA) + (ACCB) → ACCA	H, N, Z, V, C
1X YY1 001	2/3	2/5	ADCX	Add accumulator and memory with carry	(ACCX) + (C) + (M) → ACCX	H, N, Z, V, C
01 0X0 011	1	2	COMX	Complement accumulator (one's)	$\overline{(ACCX)}$ → ACCX	N, Z, V, C
01 1Z0 011	2/3	6/7	COM	Complement memory (one's)	(M) → M	N, Z, V,
01 0X0 000	1	2	NEGX	Complement accumulator (two's)	− (ACCX) → ACCX	N, Z, V, C
01 1Z0 000	2/3	6/7	NEG	Complement memory (two's)	− (M) → M	N, Z, V, C
1X YY0 000	2/3	2/5	SUBX	Subtract memory from accumulator	(ACCX) − (M) → ACCX	N, Z, V, C
00 010 000	1	2	SBA	Subtract accumulator B from accumulator A	(ACCA) − (ACCB) → ACCA	N, Z, V, C
1X YY0 010	2/3	2/5	SBCX	Subtract memory from accumulator with borrow	(ACCX) − (C) − (M) → ACCX	N, Z, V, C
01 0X1 111	1	2	CLRX	Clear accumulator	0 → ACCX	N, Z, V, C
01 1Z1 111	2/3	6/7	CLR	Clear memory	0 → M	N, Z, V, C
01 0X1 010	1	2	DECX	Decrement accumulator	(ACCX) − 1 → ACCX	N, Z, V
01 1Z1 010	2/3	6/7	DEC	Decrement memory	(M) − 1 → M	N, Z, V
01 0X1 100	1	2	INCX	Increment accumulator	(ACCX) + 1 → ACCX	N, Z, V
01 1Z1 100	2/3	6/7	INC	Increment memory	(M) + 1 → M	N, Z, V
00 011 001	1	2	DAA	Decimal adjust accumulator A	Addition des Korrekturfaktors 0110_2: wenn $b_3 b_2 b_1 b_0$ > 1001, ∨ H = 1 oder $b_7 b_6 b_5 b_4$ > 1001, ∨ C = 1	N, Z, V, C
1X YY0 100	2/3	2/5	ANDX	AND accumulator with memory	(ACCX) ∧ (M) → ACCX	N, Z, V
1X YY1 000	2/3	2/5	EORX	Exclusive OR accumulator with memory	(ACCX) ⊻ (M) → ACCX	N, Z, V
1X YY1 010	2/3	2/5	ORAX	Inclusive OR accumulator with memory	(ACCX) ∨ (M) → ACCX	N, Z, V

Machinecode	Words	Cycles ≡ States	Mnemonic	Verklaring	Schrijfwijze	Status-register
1X YY0 101	2/3	2/5	BITX	Bit test	(ACCX) ∧ (M)	N, Z
1X YY0 001	2/3	2/5	CMPX	Compare accumulator with memory	(ACCX) − (M)	N, Z, V, C
00 010 001	1	2	CBA	Compare accumulator A with accumulator B	(ACCA) − (ACCB)	N, Z, V, C
01 0X1 101	1	2	TSTX	Test accumulator	(ACCX) − 0	N, Z, V, C
01 1Z1 101	2/3	6/7	TST	Test memory	(M) − 0	N, Z, V, C
00 001 100	1	2	CLC	Clear carry	0 → C	C
00 001 101	1	2	SEC	Set carry	1 → C	C
00 001 010	1	2	CLV	Clear overflow	0 → V	V
00 001 011	1	2	SEV	Set overflow	1 → V	V
00 001 110	1	2	CLI	Clear interrupt mask	0 → I	I
00 001 111	1	2	SEI	Set interrupt mask	1 → I	I
00 000 110	1	2	TAP	Transfer contents of accumulator A to status register	(ACCA) ∨ 11 000 000 → CCR	$b_5 b_4 b_3 b_2$, $b_1 b_0$
00 000 111	1	2	TPA	Transfer contents of status register to accumulator A	(CCR) ∨ 11 000 000 → ACCA	$b_5 b_4 b_3 b_2$, $b_1 b_0$
1X YY0 110	2/3	2/5	LDAX	Load accumulator	(M) → ACCX	N, Z, V
1X YY0 111	2/3	4/6	STAX	Store accumulator	(ACCX) → M YY ≠ 00	N, Z, V
00 010 110	1	2	TAB	Transfer contents of accumulator A to accumulator B	(ACCA) → ACCB	N, Z, V
00 010 111	1	2	TBA	Transfer contents of accumulator B to accumulator A	(ACCB) → ACCA	N, Z, V
01 0X1 001	1	2	ROLX	Rotate left accumulator	(A_m) → A_{m+1} (A_7) → C (C) → A_0	N, Z, V*, C
01 1Z1 001	2/3	6/7	ROL	Rotate left memory		N, Z, V*, C
01 0X0 110	1	2	RORX	Rotate right accumulator	(A_{m+1}) → A_m (A_0) → C (C) → A_7	N, Z, V*, C
01 1Z0 110	2/3	6/7	ROR	Rotate right memory		N, Z, V*, C
01 0X1 000	1	2	ASLX	Arithmetic shift left accumulator	(A_m) → A_{m+1} (A_7) → C 0 → A_0	N, Z, V*, C
01 1Z1 000	2/3	6/7	ASL	Arithmetic shift left memory		N, Z, V*, C

2.2 6800/6809

Machinecode	Words	Cycles = States	Mnemonic	Verklaring	Schrijfwijze	Status-register
01 0X0 100	1	2	LSRX	Logical shift right accumulator	$(A_{m+1}) \rightarrow A_m$ $(A_7) \rightarrow A_6$ $0 \rightarrow A_7$ $(A_0) \rightarrow C$	N, Z, V*, C
01 1Z0 100	2/3	6/7	LSR	Logical shift right memory		N, Z, V*, C
01 0X0 111	1	2	ASRX	Arithmetic shift right accumulator	$(A_{m+1}) \rightarrow A_m$ $(A_7) \rightarrow A_7$ $(A_7) \rightarrow A_6$ $(A_0) \rightarrow C$	N, Z, V*, C
01 1Z0 111	2/3	6/7	ASR	Arithmetic shift right memory		N, Z, V*, C
10 YY1 100	2/3	3/6	CPX	Compare index register	$(IX_L) - (M+1)$ $(IX_H) - (M)$	N, Z, V
00 001 001	1	4	DEX	Decrement index register	$(IX) - 1 \rightarrow IX$	Z
00 110 100	1	4	DES	Decrement stack pointer	$(SP) - 1 \rightarrow SP$	-
00 001 000	1	4	INX	Increment index register	$(IX) + 1 \rightarrow IX$	Z
00 110 001	1	4	INS	Increment stack pointer	$(SP) + 1 \rightarrow SP$	-
11 YY1 110	2/3	3/6	LDX	Load index register	$(M) \rightarrow IX_H$ $(M+1) \rightarrow IX_L$	N, Z, V
10 YY1 110	2/3	3/6	LDS	Load stack pointer	$(M) \rightarrow SP_H$ $(M+1) \rightarrow SP_L$	Z, V
11 YY1 111	2/3	5/7	STX	Store index register	$(IX_H) \rightarrow M$ $(IX_L) \rightarrow M + 1$ $YY \neq 00$	N, Z, V
10 YY1 111	2/3	5/7	STS	Store stack pointer	$(SP_H) \rightarrow M$ $(SP_L) \rightarrow M + 1$	N, Z, V
00 110 101	1	4	TXS	Transfer from index register to stack pointer	$(IX) - 1 \rightarrow SP$	-
00 110 000	1	4	TSX	Transfer from stack pointer to index register	$(SP) + 1 \rightarrow IX$	-
00 110 11X	1	4	PSHX	Push data onto stack	$(ACCX) \rightarrow (SP)$ $(SP) - 1 \rightarrow SP$	-
00 110 01X	1	4	PULX	Pull data from stack	$(SP) + 1 \rightarrow SP$ $((SP)) \rightarrow ACCX$	-
00 100 000	2	4	BRA	Branch always	$(PC) + 2 + (.+1) \rightarrow PC$	
00 100 010	2	4	BHI	Branch if higher $(C \vee Z = 0)$		
00 100 011	2	4	BLS	Branch if lower or same $(C \vee Z = 1)$		
00 100 100	2	4	BCC	Branch if carry clear $(C = 0)$		
00 100 101	2	4	BCS	Branch if carry set $(C = 1)$		
00 100 110	2	4	BNE	Branch if not zero $(Z = 0)$		

* betekent:
V = (NVC), na de uitvoering van de schuifopdracht wordt V daarbij gevormd.

Machinecode	Words	Cycles = States	Mnemonic	Verklaring	Schrijfwijze	Status-register
00 100 111	2	4	BEQ	Branch if zero $(Z = 1)$	if condition is true $(PC) + 2 + (.+1) \rightarrow PC$ else $(PC) + 2 \rightarrow PC$	
00 101 000	2	4	BVC	Branch if overflow clear $(V = 0)$		
00 101 001	2	4	BVS	Branch if overflow set $(V = 1)$		
00 101 010	2	4	BPL	Branch if plus $(N = 0)$		
00 101 011	2	4	BMI	Branch if minus $(N = 1)$		
00 101 100	2	4	BGE	Branch if greater or equal to zero $(N \vee V = 0)$		
00 101 101	2	4	BLT	Branch if less than zero $(N \vee V = 1)$		
00 101 110	2	4	BGT	Branch if greater than zero $(Z \vee (N \vee V) = 0)$		
00 101 111	2	4	BLE	Branch if less than or equal to zero $(Z \vee (N \vee V) = 1)$		
01 1Z1 110	2/3	3/4	JMP	Jump	$(.+1) + (IX) \rightarrow PC$ $(.+2)\,(.+1) \rightarrow PC$	
10 101 101	2	8	JSR	Jump to subroutine	$(PC) + 2 \rightarrow PC$ $(PC_L) \rightarrow (SP)$ $(PC_H) \rightarrow (SP) - 1$ $(SP) - 2 \rightarrow SP$ $(.+1) + (IX) \rightarrow PC$	-
10 111 101	3	9	JSR	Jump to subroutine	$(PC) + 3 \rightarrow PC$ $(PC_L) \rightarrow (SP)$ $(PC_H) \rightarrow (SP) - 1$ $(SP) - 2 \rightarrow SP$ $(.+2)\,(.+1) \rightarrow PC$	-
00 111 001	1	5	RTS	Return from subroutine	$((SP)+1) \rightarrow PC_H$ $((SP)+2) \rightarrow PC_L$ $(SP) + 2 \rightarrow SP$	-
10 001 101	2	8	BSR	Branch to subroutine	$(PC) + 2 \rightarrow PC$ $(PC_L) \rightarrow (SP)$ $(PC_H) \rightarrow (SP)$ $(SP) - 2 \rightarrow SP$ $(.+1) + PC \rightarrow PC$	-
00 111 111	1	12	SWI	Software interrupt	$(PC) + 1 \rightarrow PC$ $(PC_L) \rightarrow (SP)$ $(PC_H) \rightarrow (SP) - 1$ $(IX_L) \rightarrow (SP) - 2$ $(IX_H) \rightarrow (SP) - 3$ $(ACCA) \rightarrow (SP) - 4$ $(ACCB) \rightarrow (SP) - 5$ $(CCR) \rightarrow (SP) - 6$ $(SP) - 7 \rightarrow SP$ $1 \rightarrow I$ $(177772)\,(177773) \rightarrow PC$	I
00 111 011	1	10	RTI	Return from interrupt	$((SP)+1) \rightarrow CCR$ $((SP)+2) \rightarrow ACCB$ $((SP)+3) \rightarrow ACCA$ $((SP)+4) \rightarrow IX_H$ $((SP)+5) \rightarrow IX_L$ $((SP)+6) \rightarrow PC_H$ $((SP)+7) \rightarrow PC_L$ $(SP) + 7 \rightarrow SP$	H, I, N, Z, V, C
00 111 110	1	9	WAI	Wait for interrupt	$(PC) + 1 \rightarrow PC$ $(PC_L) \rightarrow (SP)$ $(PC_H) \rightarrow (SP) - 1$ $(IX_L) \rightarrow (SP) - 2$ $(IX_H) \rightarrow (SP) - 3$ $(ACCA) \rightarrow (SP) - 4$ $(ACCB) \rightarrow (SP) - 5$ $(CCR) \rightarrow (SP) - 6$ $(SP) - 7 \rightarrow SP$ $(177770)\,(177771) \rightarrow PC$ oder $(177774)\,(177775) \rightarrow PC$	-
00 000 001	1	2	NOP	No operation	$(PC) + 1 \rightarrow PC$	-

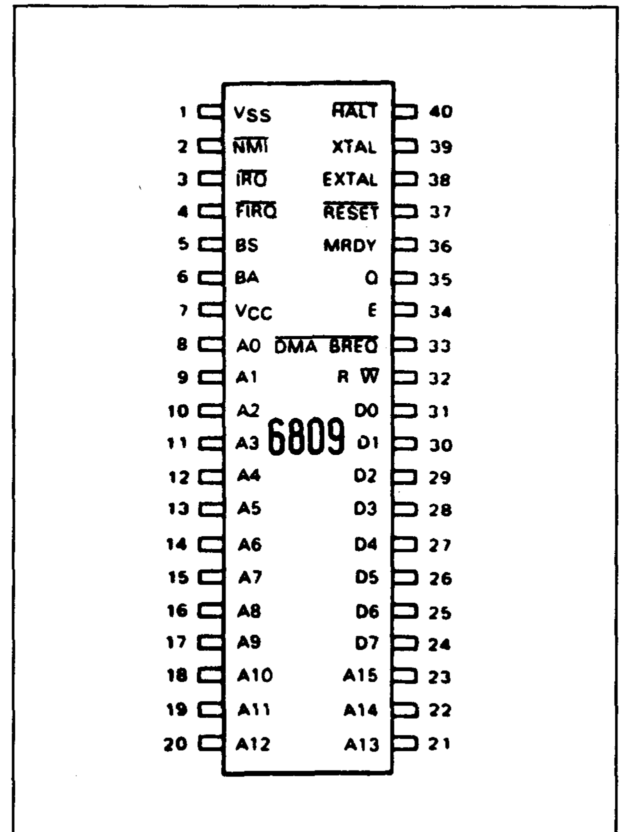
Tabel 7/2.2-1: Instructie-set van de 6800.

2.2 6800/6809

Voor de aanpassing aan nieuwe eisen volgden de processortypen 6802, met geïntegreerde klokgenerator en 256 byte RAM en daarop als nieuwste ontwikkeling, de 6809, met de volgende kenmerken:

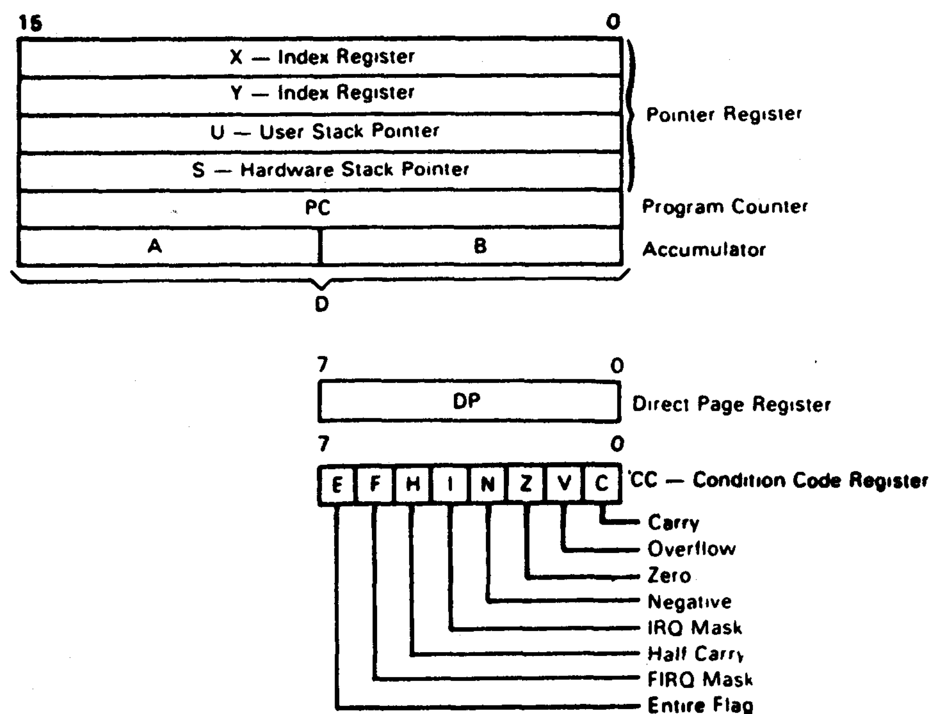
- software compatibel met 6800 en 6802.
- bruikbaar met alle 68XX periferie-eenheden.
- twee 16-bit brede index-registers.
- twee 16-bit brede te indicieren stapel-aanwijzer.
- 8 bit pagina-register voor direkte sprong adressering.
- zeer snelle interrupt (FIRQ), plaatst alleen het status-register op de stapel (stack).
- DMA/BREQ-ingang maakt het uitschakelen van de CPU mogelijk voor DMA- en geheugenverfrissingsdoeleinden.
- 92 basisinstructies en 1464 bewerkingen, inclusief de adresseerwijzen.
- 8 x 8 bit-vermeningvuldiging.
- inhoud van alle registers uitwisselbaar, of naar de stapel te verplaatsen

Figuur 7/2.2-3 toont de aanluistingen en Figuur 7/2.2-4 de uitgebreide registerbezetting. De 6809 is een van de meest uitgebreide 8-bit CPU's en biedt een omvangrijke software aanbod, waaronder de compilertalen BASIC en PASCAL.



Figuur 7/2.2-3: Aansluitgegevens van de 6809.

2.2 6800/6809



Figuur 7/2.2-4: De register-set van de 6809.

7/2.3

8080/8085

De 8085 is de verbeterde versie van de 'oude' processor 8080 van Intel. De 8-bit N-kanaal processor bevat, in tegenstelling tot zijn voorganger, extra schakelingen zoals een klokgenerator en een geïntegreerde bus-drijver. Daarnaast zijn er twee nieuwe bevelen voor een gemaskeerde interrupt-verwerking (RIM-en SIM). Deze bewerkingen maken de seriële in- en uitvoer van bitpatronen op interruptniveau mogelijk. De overige instructieset (Tabel 7/2.3-1) is software-compatibel met de 8080 processor; de klokfrequentie werd evenwel tot 3 MHz verhoogd.

Opdracht Mnemonisch	Verklaring	Op-code	Format	Bewerking	effect op vlag	cy-cli
	Dataoverdracht instructies (enkelv. registers)					
MOV r1, r2	r1, r2 = A, B, C, D, E, H of L laad r1 met inhoud van r2	01dddsss	1	(sss) → ddd	geen	5
MOV r, M	laad register met geheugeninhoud	01ddd110	1	(@ HL) → ddd		7
MOV M, r	sla registerinhoud in geheugen op	01110sss	1	(sss) → @ HL		7
MVI r	laad register met constante	00ddd110	2	(2de byte) @ ddd		7
MVI M	sla constante op	00110110	2	(2de byte) → @ HL		10
STA adr	sla inhoud van de accu op	00110010	3	(A) → (3de en 2de byte)		13
LDA adr	laad accu direct	00111010	3	((3de en 2de byte)) → A		13
STAX B	sla accu indirect op, die door register B is geadresseerd	00000010	1	(A) → @ BC		7
STAX D	sla accu indirect op, die door register D is geadresseerd	00010010	1	(A) → @ DE		7
LDAX B	laad accu indirect, die door register B is geadresseerd	00001010	1	(@ BC) → A		7
LDAX D	laad accu indirect, die door register D is geadresseerd	00011010	1	(@ DE) → A		7
	Dataoverdracht instructies (dubbele registers)					
LXI B	laad register B, geïmpliceerd met adres	00000001	3	(2de byte) → C; (3de byte) → B	geen	10
LXI D	laad register D, geïmpliceerd met adres	00010001	3	(2de byte) → E; (3de byte) → D		10
LXI H	laad register H, geïmpliceerd met adres	00100001	3	(2de byte) → L; (3de byte) → H		10
SHL D	sla register HL op onder adr. (adr.+1)	00100010	3	(L) → (3de en 2de byte) (H) → (3de en 2de byte +1)		16
LHL D	laad register HL met inhoud adr. (adr.+1)	00101010	3	((3de en 2de byte)) → L ((3de en 2de byte + 1)) → H		16
XCH G	verwissel de inhoud van de registers	11101011	1	(HL) → DE; (DE) → HL		4
	in- uitvoer opdrachten					
IN	accu wordt met de inhoud van de invoer geladen	11011011	2	(invoerkanaal in de 2de byte) → A	geen	10
OUT	inhoud accu wordt aan het uitvoer-kanaal geleverd	11010011	2	(A) → uitvoerkanaal in de 2de byte		10

Tabel 7/2.3-1: De instructie-set van de 8085.

2.3 8080/8085

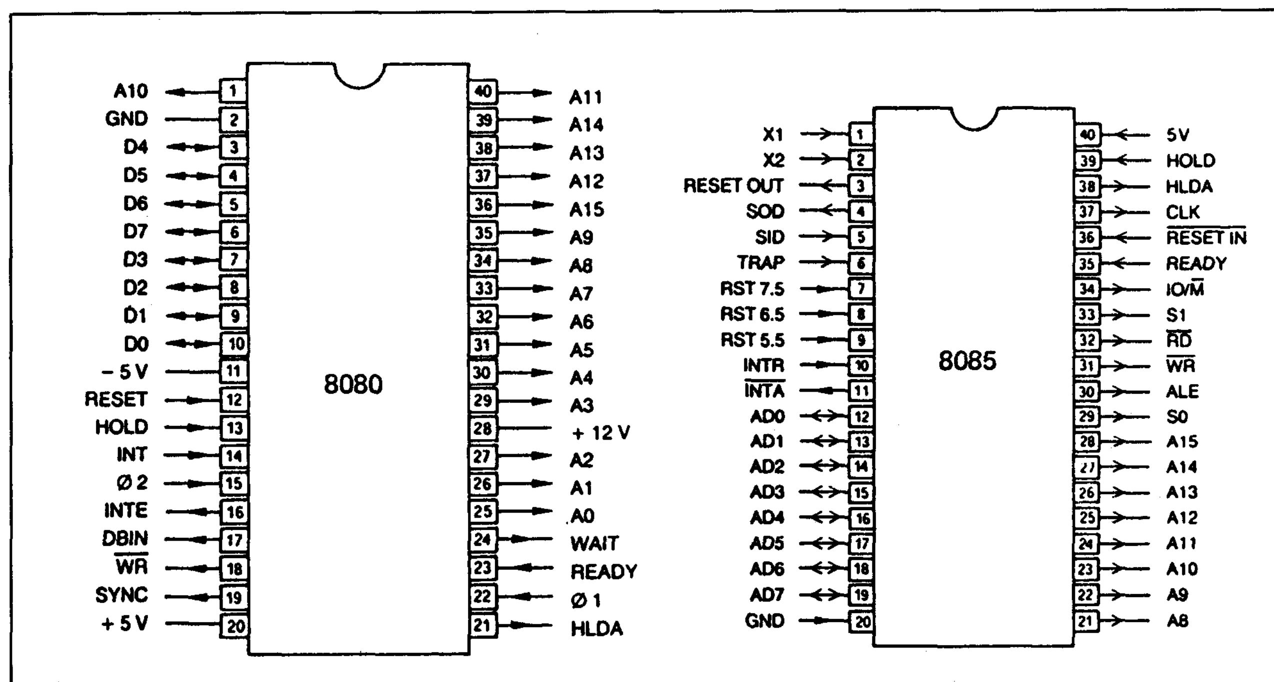
Opdracht Mnemonisch	Verklaring	Op-code	Format	Bewerking	effect op vlag	cy-cli
	rekenkundige opdrachten (enkelv. register)					
ADD r	inhoud van register 1 wordt opgeteld bij de	10000sss	1	(A) + (sss) → A	alle	4
	inhoud van de accu				vlaggen	
ADD M	geheugeninhoud wordt opgeteld bij de inhoud	10000110	1	(A) + (@ HL) → A	worden	7
	van de accu				overeen-	
ADI	constante wordt bij inhoud van de accu opgeteld	11000110	2	(A) + (2de byte) → A	komstig	7
ADC r	inhoud r1 en inhoud carry-flag worden bij de	10001sss	1	(A)+(sss)+(C) → A	de resul-	4
	accu-inhoud opgeteld				taten	
ADC M	inhoud geheugen en inhoud carry-flag worden	10001110	1	(A)+ (@ HL)+(C) →A	gezet	7
	bij de accu-inhoud opgeteld					
ACI	constante en carry-flag worden bij de accu opget.	11001110	2	(A)+(2de byte)+(C) → A	alle	7
SUB r		10010sss	1	(A)-(sss) → A	vlaggen	4
SUB M		10010110	1	(A)-(@ HL) → A	worden	7
SUI		11010110	2	(A)-(2de byte) → A	overeen-	7
SBB r	aftrekbevelen gelijk aan die voor optellen.	10011sss	1	(A)-(sss)-(C) → A	komstig	4
SBB M		10011110	1	(A)-(@ HL)-(C) → A	het re-	7
SBI		11011110	2	(A)-(2de byte)-(C) → A	sultaat	7
CMP r	inh. accu wordt vergeleken met inhoud register r1	10111sss	1	(A)-(sss) zet vlaggen	gezet	4
CMP M	inh. accu wordt met inhoud geheugen vergeleken	10111110	1	(A)-(@ HL) zet vlaggen		7
CPI	inhoud accu wordt met constante vergeleken	11111110	2	(A)-(2de byte) zet vlaggen		7
	rekenkundige en decimaal-rekenkundige opdrachten					
DAD B	inh. register B en register H worden in H opgeteld	00001001	1	(HL)+(BC) →HL	C-vlag	10
DAD D	inh. register D en register H worden in H opgeteld	00011001	1	(HL)+(DE) → HL	C-vlag	10
DAD H	inh. register H en register H worden in H opgeteld	00101001	1	(HL)+(HL) → HL	C-vlag	10
DAD SP	inh. register SP en register H worden in H opgeteld	00111001	1	(HL)+(SP) → HL	C-vlag	10
DAA	inhoud accu wordt in tweecijferig getal omgezet	00100111	1		alle	4
					vlaggen	
	logische opdrachten					
ANA r	inhoud accu en register r1 worden AND bewerkt	10100sss	1	(A) $\wedge$ (sss) → A	alle	4
ORA r	inhoud accu en register r1 worden OR bewerkt	10110sss	1	(A) $\vee$ (sss) → A	vlaggen	4
XRA r	inh. accu en reg. r1 worden Exclusive-OR bewerkt	10101sss	1	(A) $\veebar$ (sss) → A	worden	4
ANA M	inhoud accu en geheugen worden AND bewerkt	10100110	1	(A) $\wedge$ (@ HL) → A	beïn-	7
ORA M	inhoud accu en geheugen worden OR bewerkt	10110110	1	(A) $\vee$ (@ HL) → A	vloed	7
XRA M	inh. accu en geheugen worden Excl.-OR bewerkt	10101110	1	(A) $\veebar$ (@ HL) → A		7
ANI	inhoud accu wordt met constante AND bewerkt	11100110	2	(A) $\wedge$ (2de byte) → A		7
ORI	inhoud accu wordt met constante OR bewerkt	11110110	2	(A) $\vee$ (2de byte) → A		7
XRI	inh. accu wordt met constante Excl.-OR bewerkt	11101110	2	(A) $\veebar$ (2de byte) → A		7
	Roteer en schuif opdrachten					
RLC	inh. accu wordt cyclisch van 1 bit naar links versch.	00000111	1	(bit 7) → bit 0 en C	C-vlag	4
RRC	inh. accu wordt cyclisch van 1 bit naar rechts versch	00001111	1	(bit 0) → bit 7 en C	C-vlag	4
RAL	inh. accu en c-bit worden naar links verschoven	00010111	1	(bit 7) →C; (C) → bit 0	C-vlag	4
RAR	inh. accu en c-bit worden naar rechts verschoven	00011111	1	(bit 0) →C; (C) → bit 7	C-vlag	4
DAD H	register H wordt naar links geschoven	00101001	1	naar links verschuiven van	C-vlag	10
				registerpaar H, L		
	in- en decrease opdrachten					
INR r	bij inhoud register r1 wordt 1 opgeteld	00ddd100	1	(ddd)+1 → ddd	N, C, Z	5
DCR r	van inhoud register r1 wordt 1 afgetrokken	00ddd101	1	(ddd)-1 → ddd	P-vlag	5
INR M	bij inhoud geheugen wordt 1 opgeteld	00110100	1	(@ HL)+1 → @ HL	"	10
DCR M	van inhoud geheugen wordt 1 afgetrokken	00110101	1	(@ HL)-1 → @ HL	"	10
INX B	inhoud register B wordt met 1 vermeerderd	00000011	1	(BC)+1 → BC	geen	5
INX D	inhoud register D wordt met 1 vermeerderd	00010011	1	(DE)+1 → DE	beïn-	5
INX H	inhoud register H wordt met 1 vermeerderd	00100011	1	(HL)+1 → HL	vloeding	5
DCX B	inhoud register B wordt met 1 verminderd	00001011	1	(BC)-1 → BC	van	5
DCX D	inhoud register D wordt met 1 verminderd	00011011	1	(DE)-1 → DE	de	5
DCX H	inhoud register H wordt met 1 verminderd	00101011	1	(HL)-1 → HL	vlaggen	5
INX SP	inhoud register SP wordt met 1 vermeerderd	00110011	1	(SP)+1 → SP		5
DCX SP	inhoud register SP wordt met 1 verminderd	00111011	1	(SP)-1 → SP		5
	sprongopdrachten					
JMP	programma wordt op adres voortgezet	11000011	3	3de en 2de byte → PC	geen	10
	(onvoorwaardelijke sprong)				beïn-	
JNZ	bij Z = O wordt programma op adres voortgezet	11000010	3	adressen in Z = 0	vloeding	10
JZ	bij Z = 1 wordt programma op adres voortgezet	11001010	3	de 3de en Z = 1	van	10
JNC	bij C = 0 wordt programma op adres voortgezet	11010010	3	2de byte C = 0	de	10
JC	bij C = 1 wordt programma op adres voortgezet	11011010	3	worden in C = 1	vlaggen	10
JPO	bij P = 0 wordt programma op adres voortgezet	11100010	3	programma- P = 0		10

2.3 8080/8085

Opdracht Mnemonisch	Verklaring	Op-code	Format	Bewerking	effect op vlag	cy-cli
JPE	bij P = 1 wordt programma op adres voortgezet	11101010	3	teller P = 1		10
JP	bij teken-bit = 0 wordt programma op adres voortgez.	11110010	3	gebracht N = 0		10
JM	bij teken-bit = 1 wordt programma op adres voortgez.	11111010	3	wanneer: N = 1		10
PCHL	programma wordt bij adres in register H voortgez.	11101001	3	(HL) → PC		5
	subroutine en terugloop opdrachten					
CALL	programma wordt op adres voortgezet, terugkeer-adres in stack	11001101	3	(3de en 2de byte) PC	geen beïn-	17
CNZ	bij Z = 0 wordt programma op adres voortgezet	11000100	3	(3de en 2de byte) Z = 0	vloeding	17
CZ	bij Z = 1 wordt programma op adres voortgezet	11001100	3	in PC terugsprong Z = 1	van	17
CNC	bij C = 0 wordt programma op adres voortgezet	11010100	3	adres op stack C = 0	de	17
CC	bij C = 1 wordt programma op adres voortgezet	11011100	3	wanneer: C = 1	vlaggen	17
CPO	bij P = 0 wordt programma op adres voortgezet	11100100	3	P = 0		17
CPE	bij P = 1 wordt programma op adres voortgezet	11101100	3	P = 1		17
CP	bij teken = 0 wordt programma op adres voortgez.	11110100	3	N = 0		17
CM	bij teken = 1 wordt programma op adres voortgez.	11111100	3	N = 1		17
RET	programma wordt op adres voortgezet dat via SZ geadresseerd werd	11001001	1	terugspr. adr. van stack PC		10
RNZ	bij Z = 0 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11000000	1	terugsprong adres van Z = 0		11
RZ	bij Z = 1 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11001000	1	stack in programma- Z = 1		11
RNC	bij C = 0 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11010000	1	teller wanneer: C = 0		11
RC	bij C = 1 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11011000	1	C = 1		11
RPO	bij P = 0 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11100000	1	P = 0		11
RPE	bij P = 1 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11101000	1	P = 1		11
RP	bij teken = 0 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11110000	1	N = 0		11
RM	bij teken = 1 wordt progr. op adr. voortgez. dat via SZ geadresseerd werd	11111000	1	N = 1		11
	onderbrekings opdrachten					
EI	INTE-status gezet, MP kan onderbreking uitzoeken	11111011	1	na EI interrupt mogelijk	geen	4
DI	INTE-status niet gezet, MP is niet te onderbreken	11110011	1	na DI interrupt niet mogelijk	beïn-	4
RST	0 constante 7, progr. 8 x const. voortgezet onderbrekingsmaskers	11aaa111	1	(PC) → stack; aaax8 → PC	vloeding v. vlaggen	10
RIM	onderbrekingsm. en seriële ingang in accu lezen	00100000	1	SID → (7de bit A)	geen vlag	4
SIM	onderbrekingsmasker en seriële uitgang inzetten stack opdrachten	00110000	1	(7de bit A) → SOD	geen vlag	4
PUSH B		11000101	1	(BC) → stack	geen vlag	11
PUSH D	register rp1 wordt naar het door de stapelaanwijzer	11010101	1	(DE) → stack	geen vlag	11
PUSH H	aangewezen adres overgebracht	11100101	1	(HL) → stack	geen vlag	11
PUSH PSW		11110101	1	(accu, vlaggen) → stack	geen vlag	11
POP B		11000001	1	(stack) → BC	geen vlag	10
POP D	register rp1 wordt met woord geladen dat door SZ	11010001	1	(stack) → DE	geen vlag	10
POP H	geadresseerd is	11100001	1	(stack) → HL	geen vlag	10
POP PSW		11110001	1	(stack) → accu, vlaggen	alle vlaggen	10
LXI SP	laad register rp1 met adres	00110001	3	(3de en 2de byte) → SP	geen vlag	10
SPHL	laad SP met inhoud register H	11111001	1	(HL) → SP	geen vlag	5
INX SP	inhoud register rp1 wordt met 1 vermeerderd	00110011	1	(SP)+1 → SP	geen vlag	5
DCX SP	inhoud register rp1 wordt met 1 verminderd	00111011	1	(SP)-1 → SP	geen vlag	5
DAD SP	inhoud registers rp1 en H worden opgeteld;	11100011	1	(HL)+(SP) → HL	C-vlag	10
	resultaat in register H		1	(HL) met ((SP)) verwisselen	geen vlag	18
	overige opdrachten					
HLT	programma stopt	01110110	1	geen bewerking	geen vlag	7
NOP	geen bewerking	00000000	1	(A) → A	geen vlag	4
CMA	inhoud accu wordt gecomplementeerd	00101111	1	(A) → A	alle vlagg.	4
STC	zet carry-flag op 1	00110111	1	1 → C-vlag	C-vlag	4
CMC	gecomplementeerde carry-flag	00111111	1	(C-vlag) → carry	C-vlag	4

2.3 8080/8085

Bijzonder aan deze CPU is het gemeenschappelijk gebruik van de onderste acht adresleidingen met de databus en de separate IOM (in-/uitvoer-geheugen) adressering, waarmee het adresseringsgebied voor het geheugen 64K plus 256 I/0-adressen wordt uitgebreid. Voor deze funkties zijn speciale stuursignalen beschikbaar (zie figuur 7/2.3-1): pen 30: Adres Latch Enable en pen 34: IO/M-select.



Figuur 7/2.3-1: Aansluitgegevens van de 8080/8085.

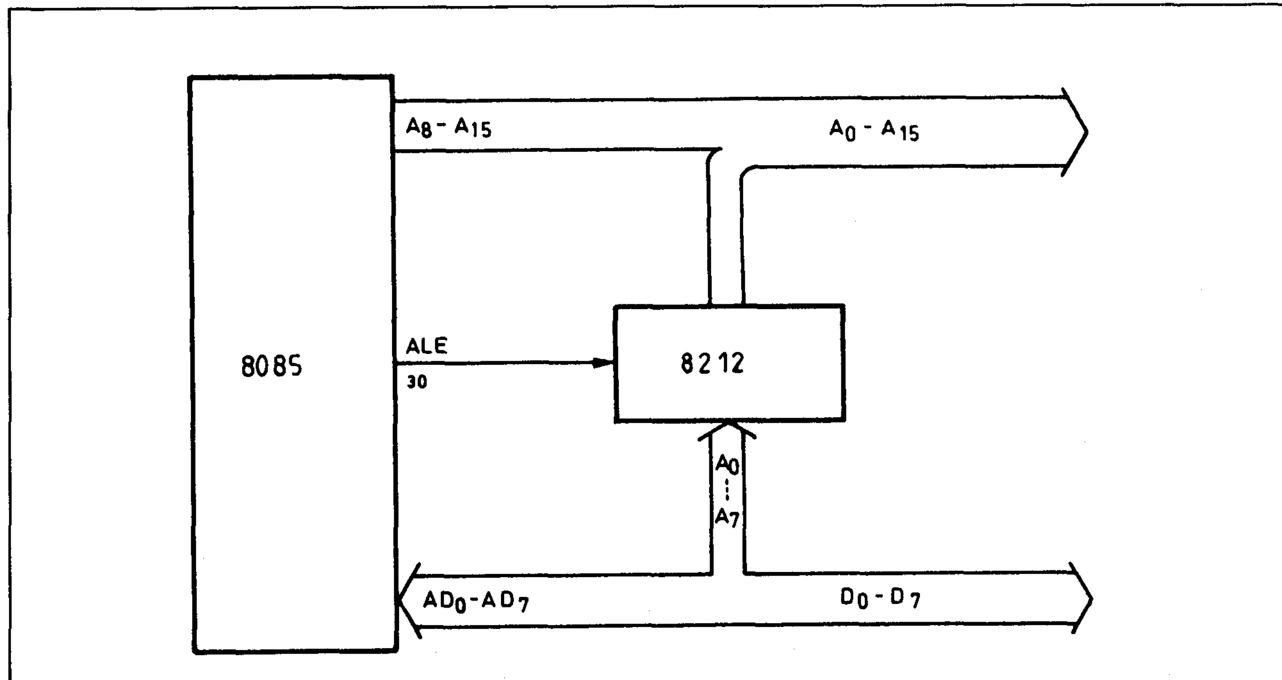
2.3 8080/8085

Aansluiting	Funktie	Toelichting
A8 tot A15	Adresbus uitgang	Adressering van de aparte geheugenbytes, hoogwaardiger bytes.
ADØ tot AD7	bi-directionele adres- en databus verbindingen	Bij de start van een gegevensoverdracht loopt de lager gewaardeerde adresbyte via deze verbindingslijnen. Gedurende de overige tijd parallelle overdracht van gegevens tussen processor en de geheugens, resp. de I/0 eenheden.
ALE	Adresbuffer uitgang	Wordt met de eerste klokperiode van een machinecyclus gezet en kunnen de adresbits AØ t.e.m. A7 opgeslagen worden.
IO/M	sturings-uitgang	Omschakeling van gecombineerde I/O geheugeneenheden van I/O-bedrijf naar geheugen-bedrijf en omgekeerd.
SØ, S1	sturings-uitgangen	Statusfuncties (vergl. tabel 4.5.1).
RD	sturings-uitgang	Statusaanduiding, Laag actief voor leescyclus van MP.
WR	sturings-uitgang	Statusaanduiding, Laag actief voor schrijfcyclus van MP.
READY	sturings-ingang	Is deze ingang laag, dan wacht de centrale eenheid totdat deze hoog wordt (toepassing voor 'single step'-bedrijf).
HOLD	sturings-ingang	Een andere eenheid gebruikt de adres- en databus. Na overgang naar Hoog-niveau geeft de centrale eenheid - na beëindiging van de lopende bewerking - de bus vrij. De MP wordt pas uit deze (hoogohmige) toestand omgezet als een Laag-niveau aanwezig is.
HLDA	sturings-uitgang	Als bevestigingssignaal wordt daarmee de HOLD-toestand aangegeven.
INTR	sturings-ingang	Interrupt voorwaarde. Wordt alleen tijdens de voorlaatste klokperiode van een opdracht nagegaan en tijdens de toestanden HOLD en HALT. Kan via programma bepaald worden. Door RESET of een interrupt-actie wordt functie gesperd.

Aansluiting	Funktie	Toelichting
INTA	sturings-uitgang	Bevestiging van een interrupt (wordt in plaats van RD gebruikt).
RST 5.5 6.5 7.5	sturings-ingang	Interruptvoorwaarde als INTR veroorzaakt interne RESTART opdracht naar opgegeven adres. Prioriteit overeenkomstig waardering.
TRAP	sturings-ingang	Interruptvoorwaarde met de hoogste prioriteit, niet via programma maskeerbaar. De ingang wordt tegelijk met de andere interrupt-ingangen uitgelezen en door geen andere interrupt opheffingsopdracht beïnvloed.
RESET IN	sturings-ingang	Met L-niveau wordt programmateller op nul gezet en het interrupt-masker evenals de HLDA-flipflop gereset. De ingang werkt asynchroon, daarom kunnen interne registers en vlaggen ongedefinieerd veranderd worden. Data- en adresbus worden hoogohmig geschakeld.
RESET OUT	sturings-uitgang	
X1, X2	ingangen	Voor kwarts, RC- of LC-combinatie voor interne klok. De werkfrequentie wordt door de klokgenerator gehalveerd en staat de processor als klokfrequentie ter beschikking.
CLK	uitgang	Klokfrequentie, bijvoorbeeld voor systeempuls.
SID	ingang	Statusfunctie voor de zevende bit van de accumulator, bij uitvoering van een RIM-opdracht.
SOD	uitgang	Statusfunktie voor de zevende bit van de accumulator bij uitvoering van een SIM-opdracht
V_{cc}	aansluiting	Voedingsspanning +5 Volt.
V_{ss}	aansluiting	0 Volt of massa.

Tabel 7/2.3-2: Functies aan de pennen van de 8085.

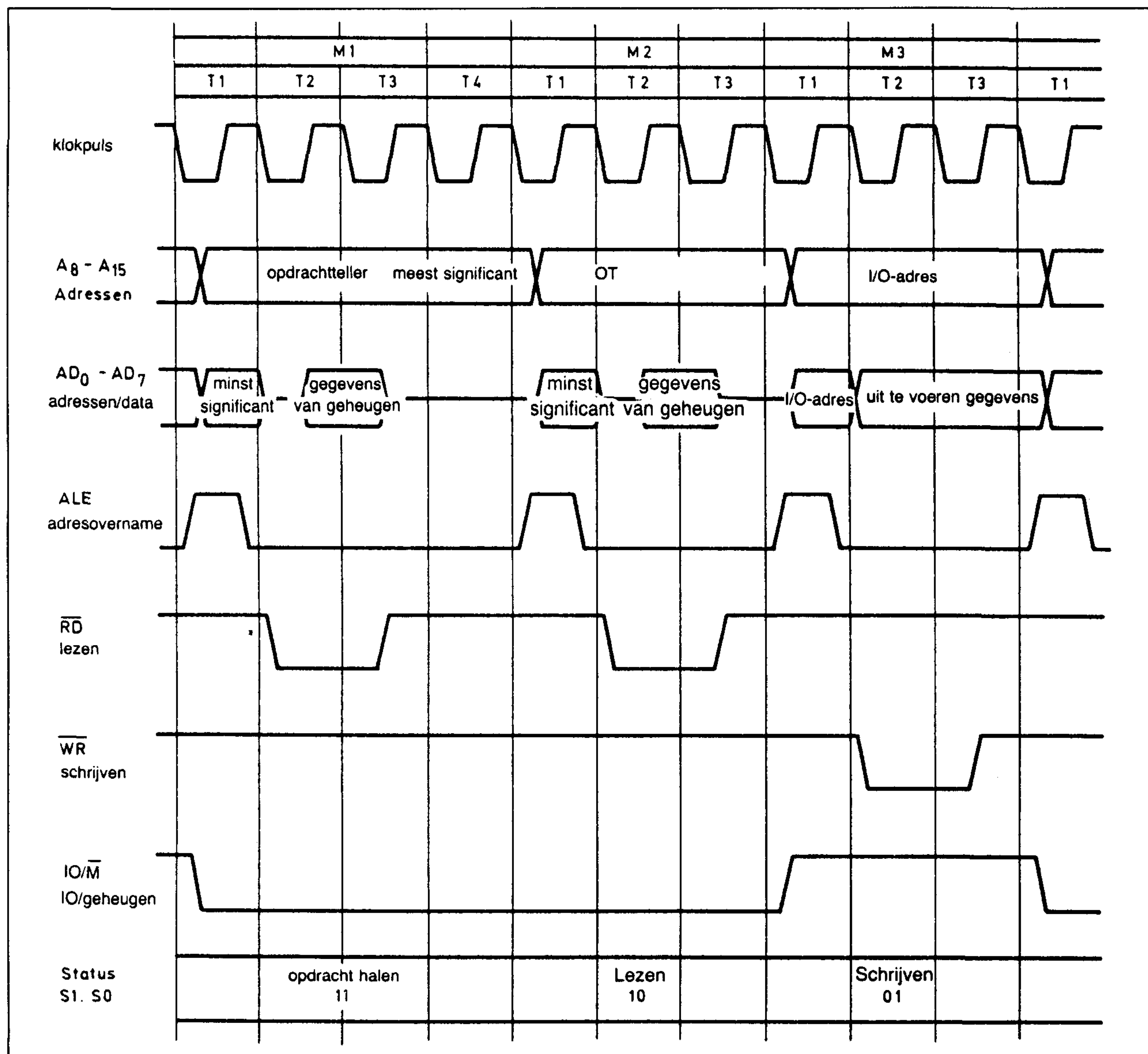
2.3 8080/8085



Figuur 7/2.3-2: Adressenopvangschakeling voor de opslag van de minst significante bits.

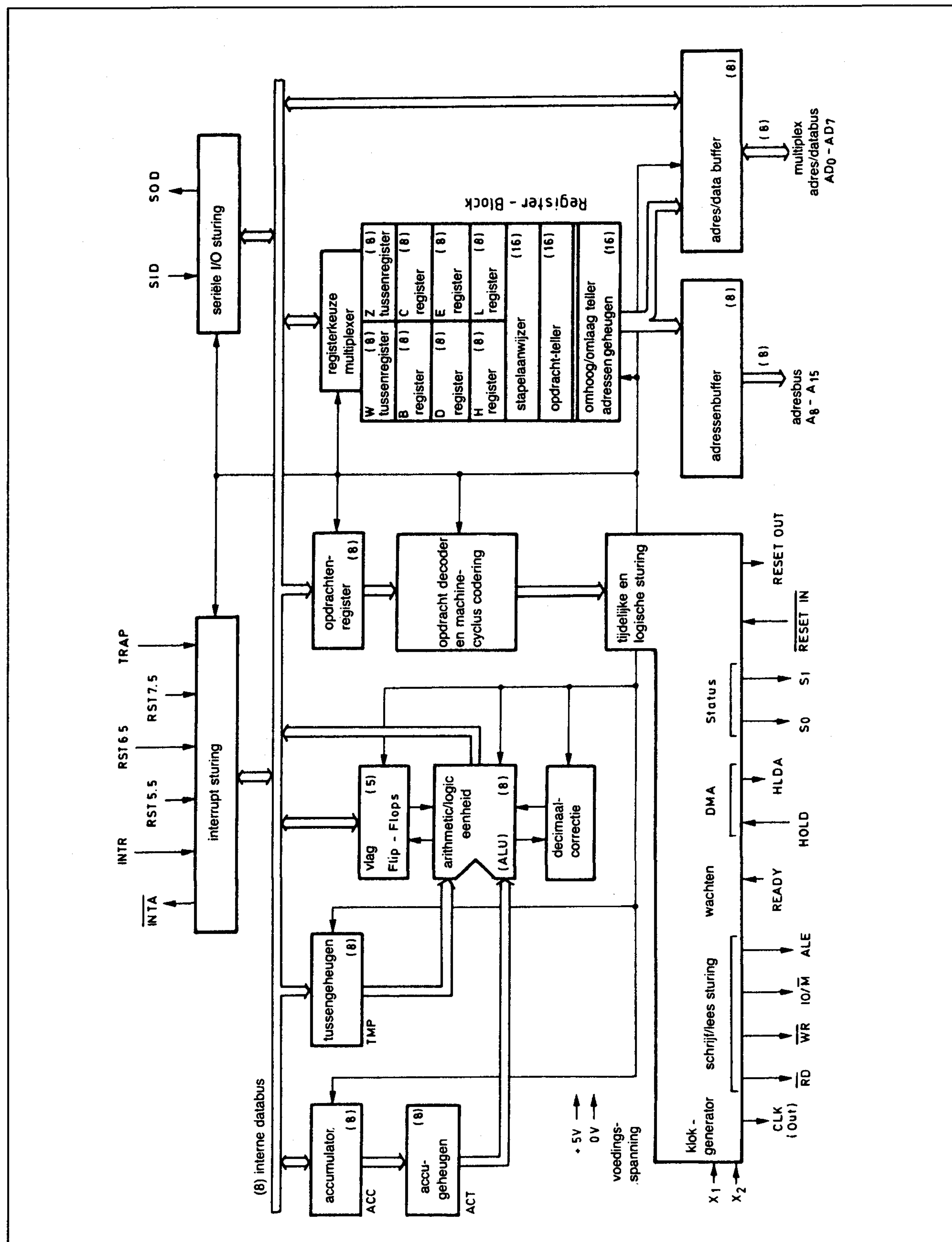
In figuur 7/2.3-2 zien we de externe schakeling voor het opvangen van de minst belangrijke 8 adres-bits, met behulp van de 8212. In het Timing-diagramg (zie figuur 7/2.3-3) is te zien op welk tijdstip de verschillende functies actief zijn. Door deze schakeltecniek blijven aansluitingen voor andere functies vrij, bijvoorbeeld voor de sturing van meerdere hardware-interrupts op de aansluitingen RST 5.5 (pen 9), RST 6.5 (pen 8) en RST 7.5 (pen 79), die de 8085 naar verschillende onderbrekings-routines kunnen laten springen. Het 3 MHz kwarts-element kan direkt op de aansluitingen X1 en X2 aangesloten worden. De CPU heeft een voedings-spanning van +5 Volt nodig en trekt 170 mA stroom. De 80XX reeks bevat vele intelligente interface-eenheden en daarnaast is er een uitgebreide keuze aan software, zoals Macro-Assembler, PL/M en CP/M.

2.3 8080/8085



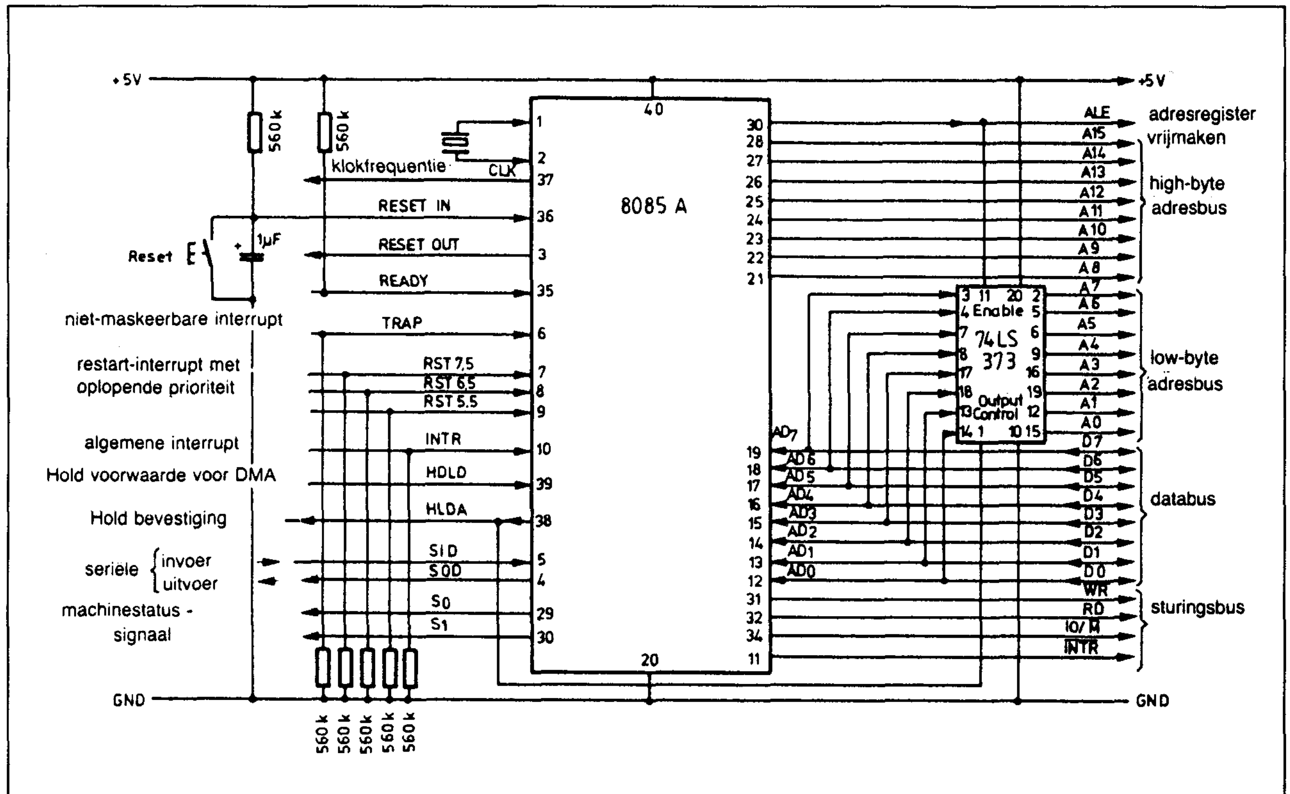
Figuur 7/2.3-3: De timing van de 8085.

2.3 8080/8085



Figuur 7/2.3-4: Blokschema van de 8085/8080.

2.3 8080/8085



Figuur 7/2.3-5: Voorbeeld-schakeling van de opbouw van een complete centrale verwerkings-eenheid met de 8085.

2.3 8080/8085

7/2.4

8088

Algemene gegevens

Versies

Intel: 8088, 8088-2, 80C88, 80C88-2
Siemens: SAB 8088, SAB 8088-2, SAB 8088-1
AMD: 8088, 8088-2, 8088-1
Fujitsu: MBL 8088, MBL 8088-2
NEC: uPD 8088

Inleiding

De 8088 kan worden beschouwd als een industriestandaard microprocessor en vormt bijvoorbeeld de basis van de beroemde IBM-PC en de meeste hieraan gelijke 'klonen'. Omdat de 8088 zo veelvuldig wordt toegepast, wordt deze microprocessor hier uitvoerig behandeld.
De 8088 is de pseudo-16 bit versie van de 8086 en heeft zowel 8 als 16 bits attributen. De 8088 kan werken in twee modes: MINnimum voor kleine systemen en MAXimum voor grotere toepassingen zoals multiprocessing.

De belangrijkste kenmerken van de 8088 zijn:
- 8-bit data bus interface;
- 16-bit interne architectuur;
- directe adressering tot 1 Mbyte geheugen;
- direct software compatibel met de 8086;
- 14 woorden van 16-bit registerset met symmetrische werking;
- 24 operand adresseermodes;
- byte, woord en blok handelingen mogelijk;
- 8-bit en 16-bit rekenen (met en zonder teken, binair of decimaal, inclusief vermenigvuldigen en delen);
- verkrijgbaar in N-kanaals MOS en CMOS uitvoering;
- verschillende kloksnelheden:
  5 MHz voor 8088 (80C88)
  8 MHz voor 8088-2 (80C88-2)
  10 MHz voor 8088-1
- 40-pens DIL-verpakking (zie figuur 7/2.4-1), CMOS ook in 44-pens PLCC.

Functionele beschrijving

Algemene werking

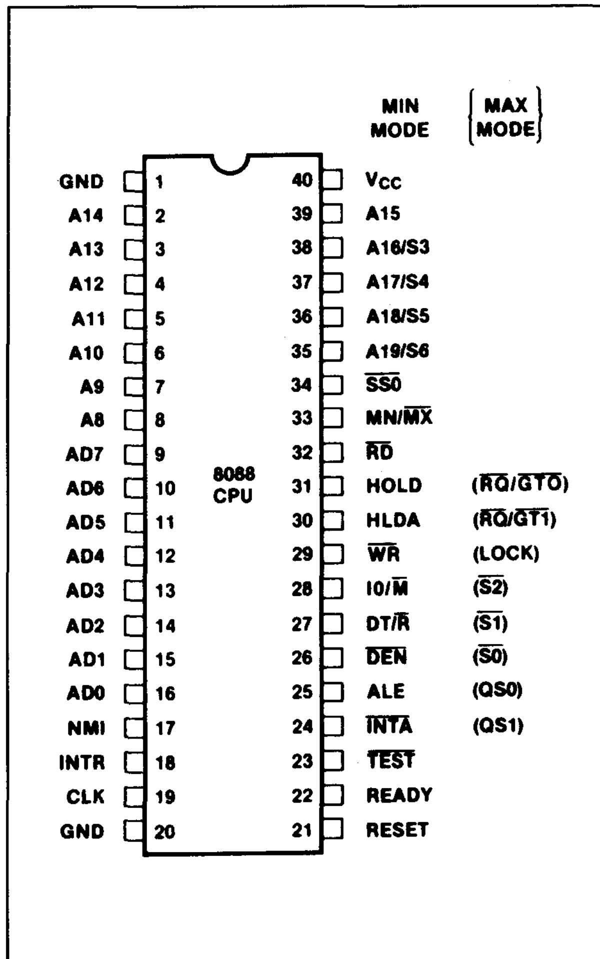
De interne functies van de 8088 microprocessor kunnen worden verdeeld in twee hoofdgroepen:
- Execution/Control Unit (EU);
- Bus Interface Unit (BIU).

In figuur 7/2.4-2 zijn de onderlinge betrekkingen tussen deze twee eenheden te zien plus de functionele delen waaruit zij zijn opgebouwd. Deze delen werken direct samen, maar vervullen hun eigen speciale functies onafhankelijk van elkaar.

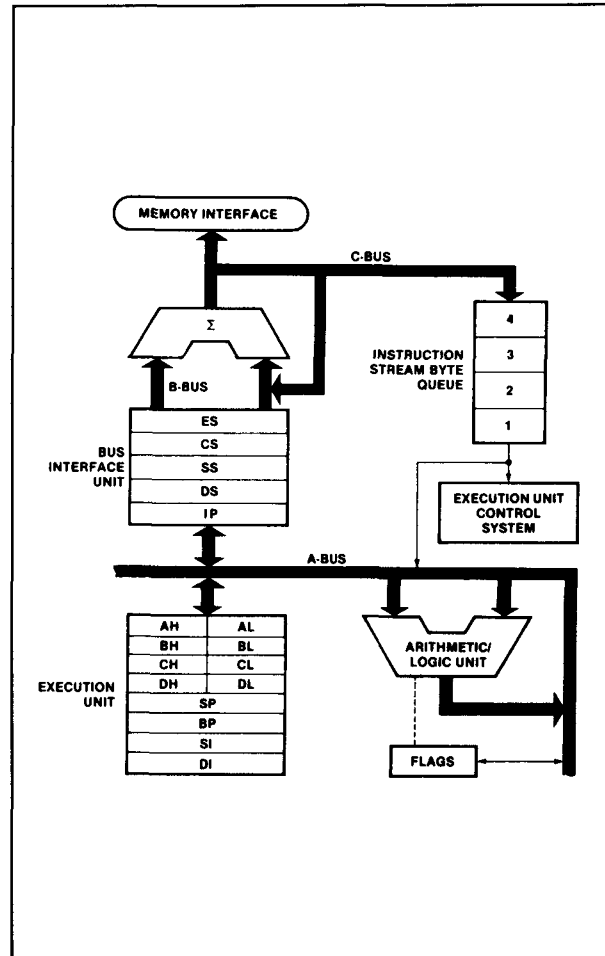
Execution Unit

De uitvoerende eenheid EU verzorgt de basis-verwerkingsfuncties, aangezien de dataregisters en de rekeneenheid (ALU) zich hier bevinden. Instructies die door de BIU zijn opgehaald, worden hier geaccepteerd terwijl niet-verplaatste operand-adressen weer terug gaan naar de BIU. Daarna ontvangt de EU geheugen-operands via de BIU, verwerkt

2.4 8088



Figuur 7/2.4-1: Aansluitgegevens van de 8088 processor.



Figuur 7/2.4-2: Principiële interne structuur van de 8088.

ze en geeft de resultaten aan de BIU voor opslag.

Bus Interface Unit

De taak van de BIU is het gebruik van de busbandbreedte zo groot mogelijk te maken (de processor-snelheid is hiervan in belangrijke mate afhankelijk). Dit wordt op twee manieren gedaan:

1 - Instructies worden opgehaald voordat de EU ze nodig heeft. De BIU buffert ze in een wachtrij die maximaal 4 bytes aan instructies kan bevatten die op decodering en uitvoering wachten. De EU behoeft daardoor niet te wachten tot een buscyclus is beëindigd om een nieuwe opdracht aan te nemen.

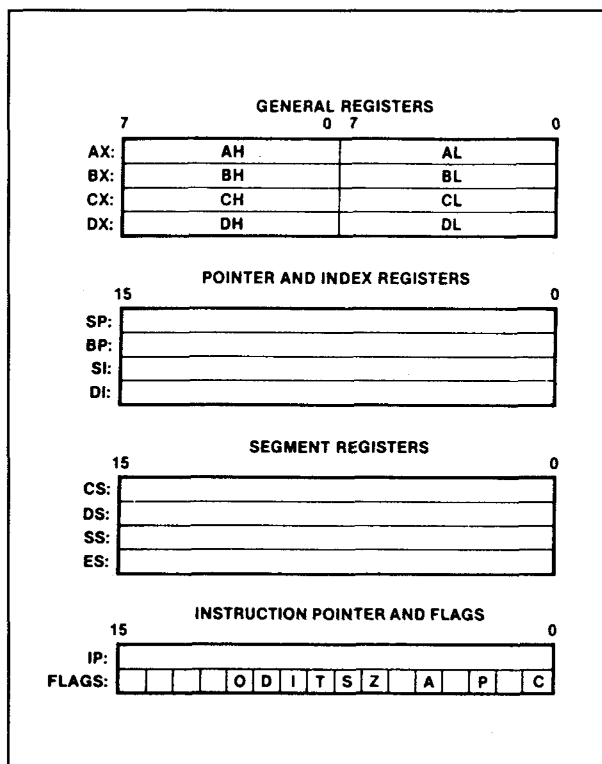
2 - De BIU voert de functies uit die te maken hebben met het ophalen en wegbergen van operands, adresverplaatsing en bus-besturing.

Registers

De 8088 bevat drie stellen van vier 16-bit registers en één stel van negen 1-bit vlaggen (zie figuur 7/2.4-3). De drie stellen registers zijn algemene registers, verwijsadressen (pointers), indexregisters en segmentregisters. Er is een 16-bit instructie-pointer die niet direct toegankelijk is maar wordt gema-

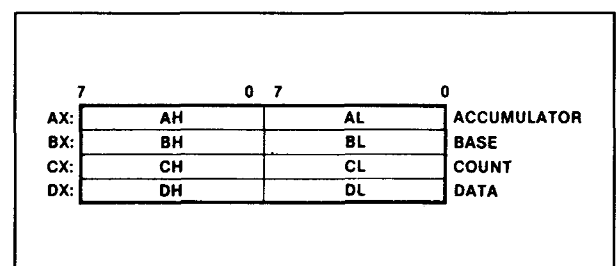
2.4 8088

de algemene register of HL-groep genoemd. Deze algemene registers kunnen onbeperkt deelnemen aan de rekenkundige en logische bewerkingen. Enkele van de overige bewerkingen van de 8088 (zoals de stringhandelingen) wijzen bepaalde algemene registers aan voor speciale toepassingen. Deze toepassingen worden gekenmerkt door de volgende mnemonische uitdrukkingen:
AX: Accumulator;
BX: Basis;
CX: tellen (Count);
DX: Data.



Figuur 7/2.4-3: Overzicht van de interne registers.

De algemene registers hebben één eigenschap die ze anders maakt dan de andere registers: de bovenste en onderste helften zijn apart adresseerbaar. De algemene registers kunnen dus worden voorgesteld als twee stellen (H en L) van vier 8-bit registers (zie ook figuur 7/2.4-4).
De accumulator onderscheidt zich op nog een andere manier: de programma's worden compacter wanneer men de accumulator als doel gebruikt van data-overdracht-, rekenkundige- en logische instructies, in plaats van de algemene registers. De overige registers in de 8088 processor kunnen niet worden onderverdeeld en moeten worden gebruikt alsof ze 16-bit woorden bevatten.

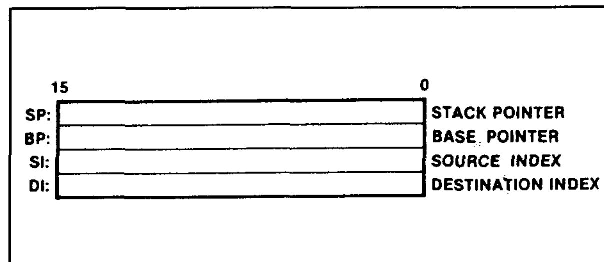


Figuur 7/2.4-4: Mogelijke opdeling van de vier algemene registers in acht 8-bit registers.

De (SP, BP, SI en DI) registergroep wordt de pointer- en indexregistergroep (P en I groep) genoemd. De registers in deze groep zijn gelijk, dit wil zeggen dat ze gewoonlijk offset-adressen bevatten die worden gebruikt om binnen een segment te adresseren. Net als de algemene registers kunnen de pointer- en indexregisters deelnemen aan de 16-bit rekenkundige en logische bewerkingen van de 8088. Ook wat betreft het deelnemen aan adres-berekeningen zijn zij gelijk. Er zijn echter enkele verschillen die ze in twee groepen verdelen, de pointergroep (SP, BP) en de indexgroep (SI, DI). Het verschil ligt in het feit dat van de pointers wordt aangenomen dat ze offset-adressen binnen het actuele stacksegment bevatten (zie ook geheugenorganisatie) en de indexen offset-adressen binnen het actuele data-segment (behalve voor string-bewerkingen). Deze systeemgekozen condities kunnen worden overstemd met een eenvoudige prefix waardoor deze registers algemeen bruikbaar worden. De bijbehorende mnemonische noteringen zijn (figuur 7/2.4-5):
SP: Stack Pointer;
BP: Basis Pointer;

2.4 8088

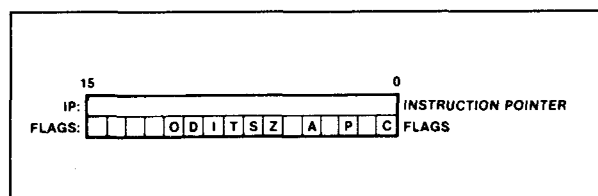
SI: bronindex (Source Index);
DI: bestemmingsindex (Destination Index).



Figuur 7/2.4-5: De pointer- en indexgroep registers.

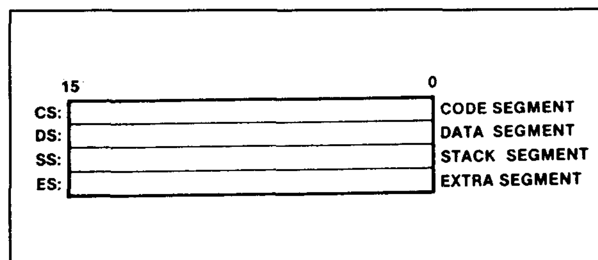
De (AF, CF, DF, OF, PF, SF, TF, ZF) registergroep wordt het vlagregister of F-groep genoemd. De vlaggen in deze groep zijn allemaal 1 bit groot en worden gebruikt om statusinformatie van de processor in op te slaan en om de werking van de processor te controleren. Het vlagregister heeft de volgende mnemonische uitdrukkingen:
AF: Auxiliary Carry;
CF: Carry;
DF: Direction;
IF: Interrupt Enable;
OF: Overflow;
PF: Parity;
SF: Sign;
TF: Trap;
ZF: Zero.

De AF, CF, PF, SF en ZF-vlaggen zijn identiek aan die in de 8080 en geven de status van de laatste rekenkundige- of logische bewerking aan. De ZF-vlag wordt gezet als het resultaat van een instructie nul is. De SF-vlag verschijnt als van een resultaat het belangrijkste bit een 1 is. Een PF-vlag die op 1 staat geeft een even pariteit aan. Een carry of borrow uit het meest significante bit zet de CF-vlag en een carry uit bit 3 naar bit 4 of een borrow uit bit 4 naar bit 3 zet de AF-vlag.
De OF-vlag is aan deze groep toegevoegd om de rekenkundige overflowconditie (met teken) te signaleren. De DF, IF en TF-vlaggen worden gebruikt om bepaalde aspecten van de processor te controleren. De DF-vlag regelt de richting van string-manipulaties (auto increment of auto decrement). De IF-vlag staat externe interrupties toe of blokkeert deze. De TF-vlag zet de processor in de stap-voor-stap mode om fouten uit programma's te halen. De vlagregisters en het formaat waarin zij door 'push flag'-handelingen worden opgeborgen zijn te zien in figuur 7/2.4-6.



Figuur 7/2.4-6: Het vlag-register met indexpointer.

De CS, DS, SS en ES registergroep wordt het segment registerbestand of S-groep genoemd (zie figuur 7/2.4-7). De segmentregisters spelen een belangrijke rol bij alle berekeningen waar geheugenadressen bij betrokken zijn. De bij het segmentregister behorende mnemonische codes zijn:
CS: Code;
DS: Date;
SS: Stack;
ES: Extra.



Figuur 7/2.4-7: De segmentregister groep.

De inhoud van het CS-register bepaalt het actuele code-segment. Van alle instructies die worden opgehaald wordt aangenomen dat zij relatief ten opzichte van CS zijn, waarbij de instructie-pointer (IP) als offset wordt

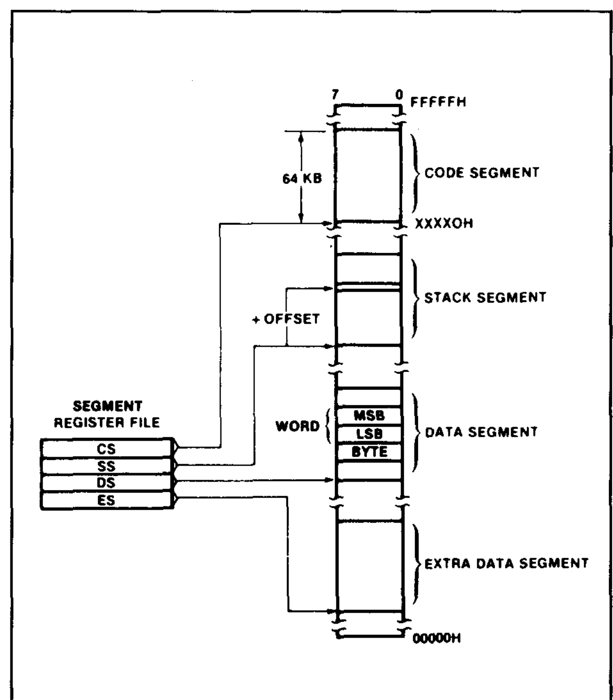
2.4 8088

gebruikt. De inhoud van het DS-register bepaalt het actuele data-segment. Van alle verwijzingen naar data (behalve die die betrekking hebben op BP of SP, of DI in een string-instructie) wordt vanuit de beginconditie aangenomen dat zij relatief zijn ten opzichte van DS. Verwijzingen naar data kunnen relatief ten opzichte van een van de andere segmentregisters worden gemaakt door de instructie vooraf te laten gaan door een 'segment override prefix' van 1 byte.
De inhoud van het SS-register bepaalt het actuele stack-segment. Alle dataverwijzingen die expliciet of impliciet te maken hebben met SP of BP, zijn vanuit de beginconditie relatief ten opzichte van SS. Ook alle push- en pop-operaties horen hierbij (inclusief de gevolgen van oproep-handelingen, interrupts en return operaties). Verwijzingen naar data die betrekking hebben op BP (maar niet SP) kunnen relatief worden gemaakt ten opzichte van een van de andere drie segmentregisters door een 'segment override prefix' te gebruiken.
De inhoud van het ES-register bepaalt het actuele extra segment. Dit extra segment wordt meestal behandeld als een bijgevoegd datasegment. Bij string-instructies worden verwijzingen naar data relatief ten opzichte van ES genomen.
Programma's die geen segmentregisters laden of manipuleren worden 'dynamisch verplaatsbaar' genoemd. Een dergelijk programma kan worden geïnterrumpeerd, naar een andere geheugenplaats gebracht en opnieuw gestart met andere segmentregisterwaarden.

Geheugen-organisatie

De processor voorziet elke byte geheugen van een 20-bit adres. Het geheugen is logisch georganiseerd als een lineair array van 1 miljoen bytes, geadresseerd tussen 00000(H) en FFFFF(H) (H= hexadecimaal). Het geheugen kan verder logisch worden onderverdeeld in code, data, alternatieve data en stack-segmenten tot maximaal 64 kbytes per stuk, waarbij elk segment op een 16-bit grens begint (zie figuur 7/2.4-8).

16-bit woord-operands kunnen op even of oneven adresgrenzen worden neergezet. Van adres- en data-operands wordt het minst belangrijke byte van het woord (LSB) opgeborgen in de laagste adreslocatie en het meest belangrijke byte (MSB) in het aansluitend hogere adres. De BIU voert bij 16-bit operands automatisch twee ophaal- of schrijfcyclussen uit.



Figuur 7/2.4-8: Geheugen-organisatie.

Sommige plaatsen in het geheugen zijn gereserveerd voor speciale CPU-handelingen (zie figuur 7/2.4-9). De plaatsen tussen FFFFOH en FFFFFH zijn bestemd voor bewerkingen die een jump naar de initiële systeem-initialisatie routine bevatten. Na een RESET zal de CPU altijd beginnen op adres FFFFOH, zodat daar de jump moet zijn gelokaliseerd. De plaatsen 00000H tot en met 003FFH zijn gereserveerd voor interrupt operaties. Door middel van vier-byte pointers, bestaande uit een 16-bit segmentadres en

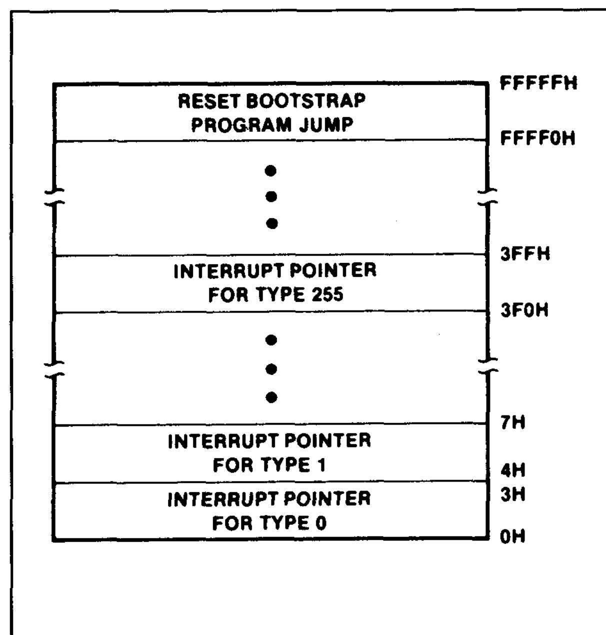
2.4 8088

een 16-bit offsetadres gaat het programma direct door naar een van de 256 mogelijke interrupt serviceroutines. Van de pointer-elementen wordt aangenomen dat zij op hun respectievelijke plaatsen in het gereserveerde geheugen waren opgeborgen, voordat de interruptie plaatsvond.

Minimum en maximum modes

De vereiste voorzieningen voor minimum en maximum 8088 systemen zijn zozeer verschillend van elkaar dat het niet mogelijk is om beide efficiënt te bedienen met 40 uniek gedefinieerde aansluitpennen. De 8088 is dan ook voorzien van een 'strap'-pen (MN/$\overline{\text{MX}}$) die de systeem-configuratie bepaalt. De definitie van een deel van de pennen verandert, afhankelijk van de toestand van de strap-pen. Ligt de MN/$\overline{\text{MX}}$-pen aan GND dan kiest de 8088 voor de pennen 24 t/m 31 en 34 de maximum mode. Is de MN/$\overline{\text{MX}}$-pen verbonden met Vcc dan genereert de 8088 zelf bus-control signalen op de pennen 24 t/m 31 en 34.

De minimum mode 8088 kan zowel met een gemultiplexte bus als een gedemultiplexte



Figuur 7/2.4-9: Gereserveerde geheugenplaatsen.

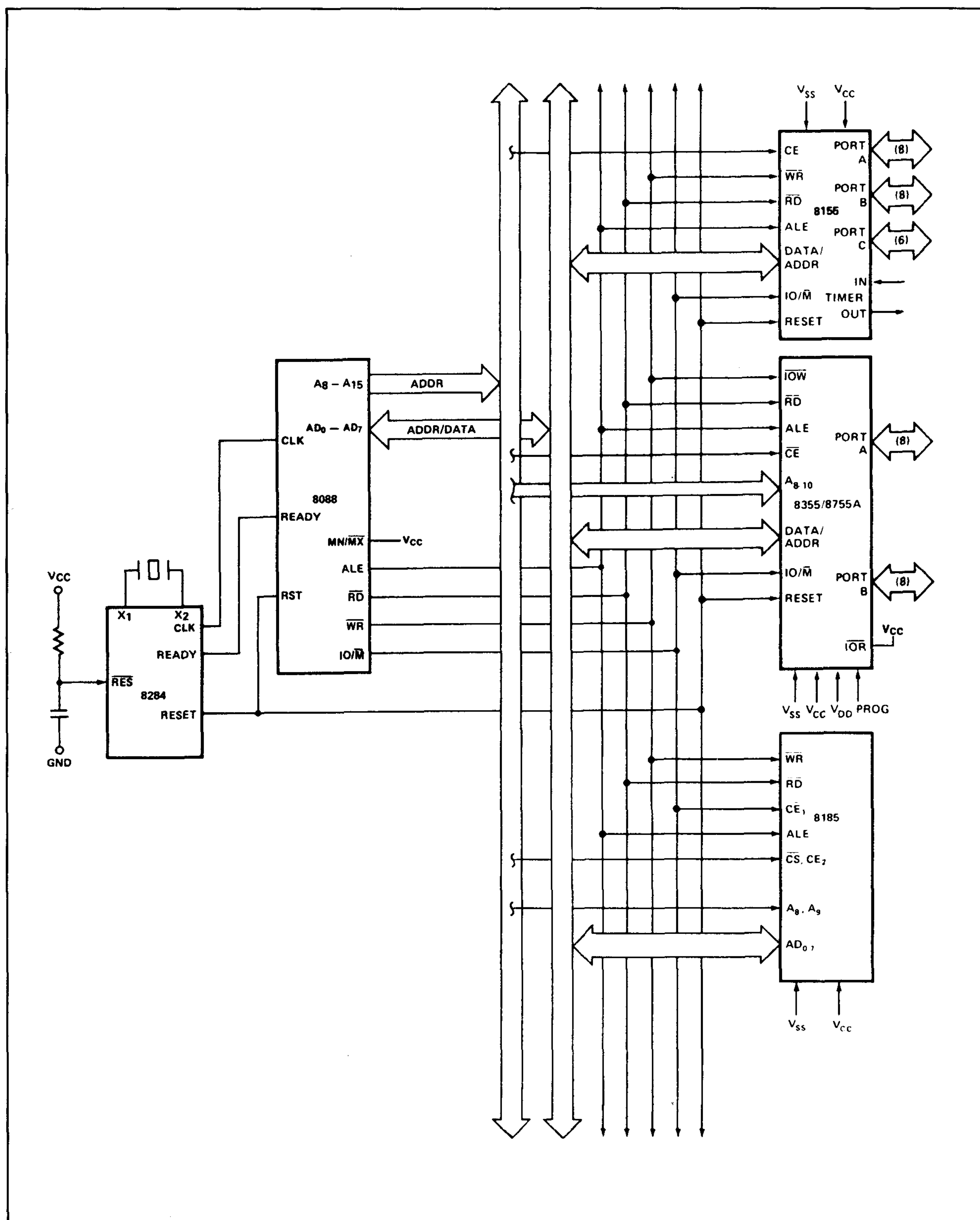
bus worden gebruikt. De gemultiplexte busconfiguratie is compatibel met de MCS-85 periferieschakelingen (8155, 8156, 8355, 8755A en 8185) met gemultiplexte bus. In deze configuratie (zie ook figuur 7/2.4-10) heeft de gebruiker een minimum aan chips nodig voor zijn systeem.
In de gedemultiplexte mode moeten extra latches worden toegepast. Met één latch kan tot 64 kB worden geadresseerd, terwijl met twee latches één volledige megabyte adresseerbaar is. Een derde latch kan als buffer worden gebruikt wanneer dit bij het laden van de adresbus nodig is. Ook kan een 8286 of 8287 transceiver nodig zijn als de databus gebufferd moet worden (figuur 7/2.4-11). De 8088 levert $\overline{\text{DEN}}$ en DT/$\overline{\text{R}}$ voor de besturing van de transceiver en ALE voor het latchen van de adressen. Deze configuratie van de minimum mode voorziet in de standaard gedemultiplexte busstructuur met zwaar gebufferde bus en minder strenge bustiming.

In de maximum mode wordt de 8288 buscontroller gebruikt (zie figuur 7/2.4-12). De 8288 decodeert de statuslijnen $\overline{\text{S0}}$, $\overline{\text{S1}}$ en $\overline{\text{S2}}$ en levert alle buscontrol-signalen die het systeem nodig heeft. Door de besturing van de bus te verplaatsen naar de 8288 kunnen de controllijnen meer stroom leveren en opnemen, terwijl de 8088-pennen worden vrijgemaakt voor uitbreiding van de mogelijkheden voor het grote systeem. In de maximum mode levert de 8088 hardware lock, queue status en twee request/grant interfaces. Door deze voorzieningen kunnen coprocessoren in lokale en niet-lokale busconfiguraties worden opgenomen.

Bus bewerkingen

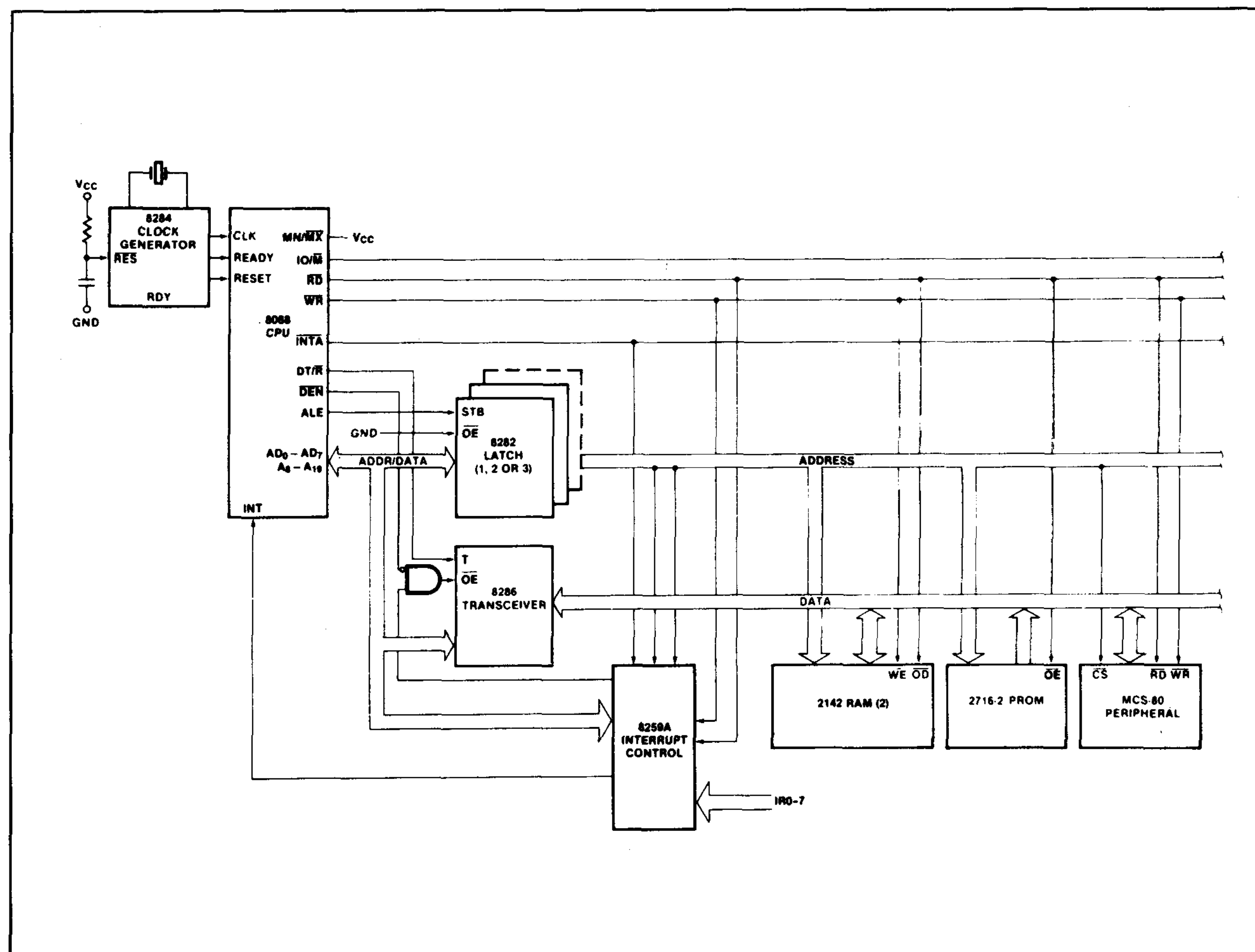
De adres/data-bus van de 8088 bestaat uit drie delen: de laagste acht adres/databits (AD0 - AD7), de middelste acht adresbits (A8 - A15) en de hoogste vier adresbits (A16 - A19). De adres/databits en de hoogste vier adresbits worden in de tijd gemultiplext. Deze techniek maakt het meest effectieve gebruik van de pennen mogelijk, waardoor een

2.4 8088



Figuur 7/2.4-10: Gemultiplexte busconfiguratie.

2.4 8088

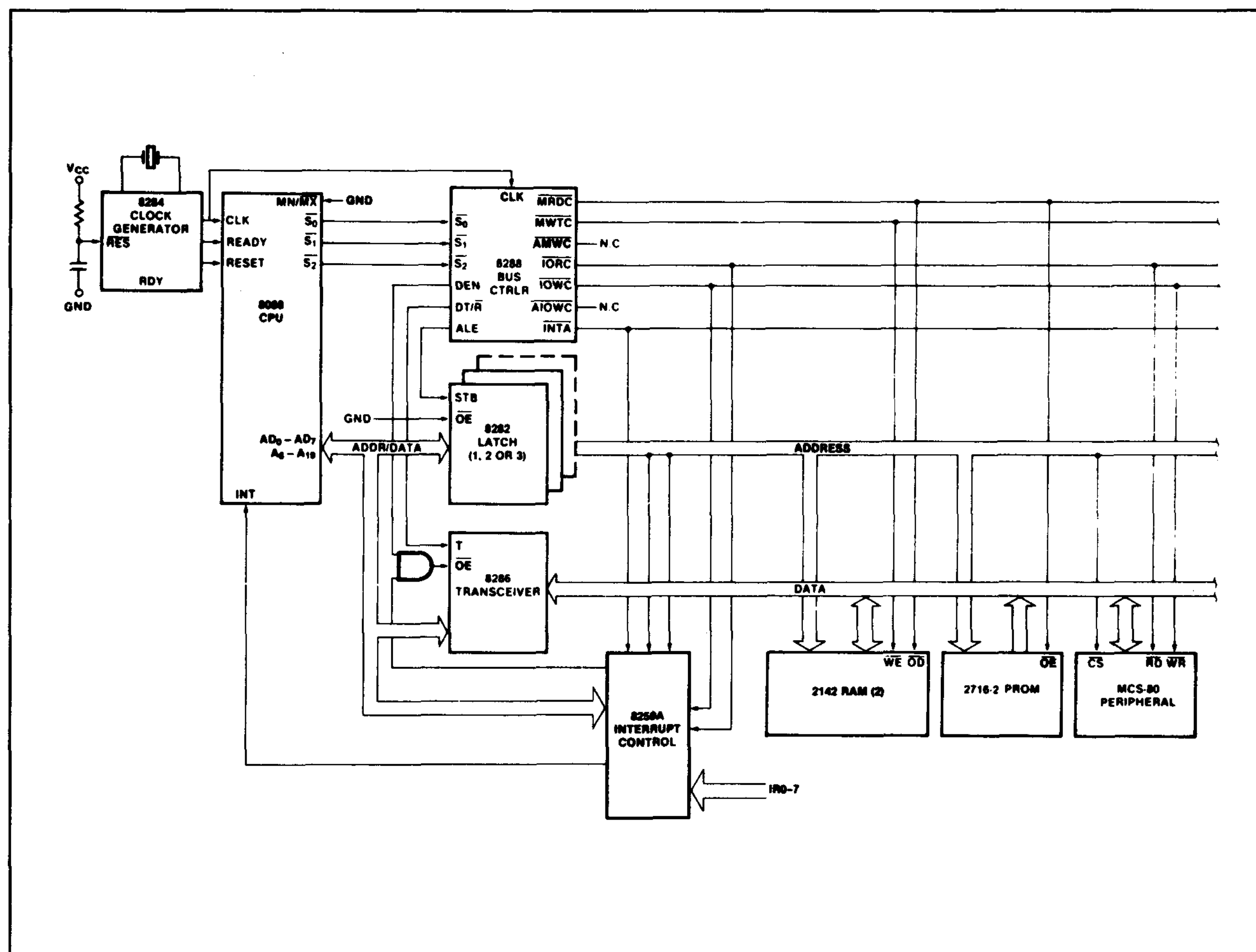


Figuur 7/2.4-11: Gedemultiplexte busconfiguratie.

standaard 40-pens behuizing bruikbaar is. De middelste acht adresbits worden niet gemultiplext maar blijven telkens gedurende de gehele buscyclus geldig. Bovendien kan de bus met een enkele adreslatch bij de processor worden gedemultiplext indien het systeem een standaard niet-gemultiplexte bus nodig heeft.
Iedere processor buscyclus bestaat uit minstens vier CLK cyclussen. Deze staan bekend als T1, T2, T3 en T4 (zie figuur 7/4.2-13). Het adres wordt gedurende T1 door de processor uitgezonden en data wordt gedurende T3 en T4 op de bus overgedragen. T2 wordt voornamelijk gebruikt om de richting van de bus te veranderen tijdens leesoperaties. In het geval dat de geadresseerde schakeling een 'NOT READY' indicatie geeft, worden wachttijden (Tw) tussen T3 en T4 geplaatst. Elke tussengevoegde wachttijd duurt even lang als een CLK-cyclus. Tussen de door de 8088 bestuurde buscyclussen kunnen perioden optreden die 'idle states' (Ti) of niet-aktieve CLK-cyclussen worden genoemd. De processor gebruikt deze cyclussen voor interne huishoudelijke zaken.

Gedurende T1 van elke buscyclus wordt het ALE-signaal (address latch enable) uitgezonden (door de processor of door de 8288 buscontroller, afhankelijk van MN/$\overline{MX}$). Op de achterflank van deze puls kunnen een geldig

2.4 8088



Figuur 7/2.4-12: Volledig gebufferd systeem waarbij gebruik gemaakt wordt van een 8288 buscontroller.

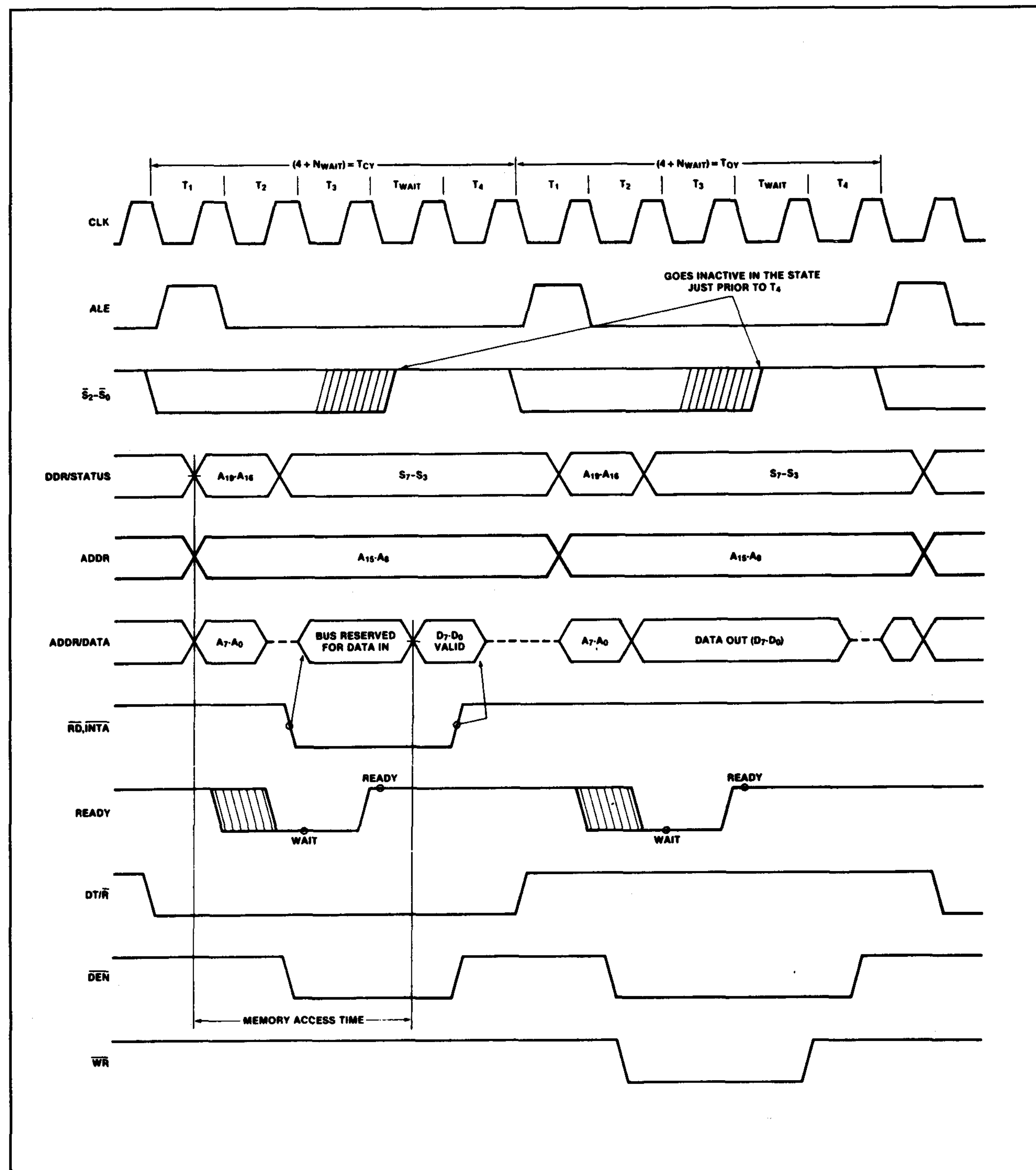
adres en bepaalde statusinformatie voor de cyclus worden gelatcht.

In de maximum mode worden de statusbits $\overline{S0}$, $\overline{S1}$ en $\overline{S2}$ door de buscontroller gebruikt om het type bustransactie aan te geven volgens tabel 7/2.4-1.
De statusbits S3 t/m S7 worden gemultiplext met de hoogste adresbits en zijn daardoor geldig gedurende T2 t/m T4. S3 en S4 geven aan welk segmentregister voor deze buscyclus werd gebruikt door het adres te vormen op de in tabel 7/2.4-2 aangegeven wijze.
S5 is een afspiegeling van het PSW interrupt enable bit. S6 is altijd gelijk aan 0 en S7 wordt niet gebruikt.

I/O adressering

In de 8088 kunnen I/O handelingen maximaal 64 kB in- en uitgangsregisters (I/O registers) adresseren. Het I/O adres verschijnt in hetzelfde formaat als het geheugenadres op de buslijnen A15 t/m A0. De adreslijnen A19 t/m A16 zijn nul bij I/O operaties. De variabele I/O instructies, die het DX-registers als pointer gebruiken, kunnen alle adressen bereiken, terwijl de directe I/O instructies rechtstreeks een of twee van de 256 I/O byte lokaties in pagina 0 van de I/O adresruimte adresseren.
I/O poorten worden op dezelfde manier geadresseerd als geheugenplaatsen.
Ontwerpers die bekend zijn met de 8085 die-

2.4 8088



Figuur 7/2.4-13: Principiële systeemtiming.

2.4 8088

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CHARACTERISTICS
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Tabel 7/2.4-1: Codering van de bustransacties met S2, S1 en S0.

S_4	S_3	CHARACTERISTICS
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

Tabel 7/2.4-2: Codering van het gebruikte segmentregister met S3 en S4.

nen te bedenken dat de 8085 I/O adresseert met een 8-bit adres op beide helften van de 16-bit adresbus. De 8088 gebruikt een compleet 16-bit adres op de laagste 16 adreslijnen.

Externe Interface

Resetten van de processor en initialisatie
De processor wordt geïnitialiseerd en start-up vindt plaats door de RESET-pen te aktiveren (HOOG). De RESET van de 8088 moet langer dan vier CLK-cyclussen HOOG zijn. De 8088 beëindigt de bewerkingen op het hoog gaan van RESET en blijft dan slapend zolang RESET HOOG is.
Bij het LAAG gaan van RESET wordt een interne reset-volgorde getriggerd die ongeveer 10 CLK-cyclussen duurt. Na deze onderbreking werkt de 8088 normaal, beginnend met de instructie in het absolute adres FFFF0H (zie figuur 7/2.4-9). De RESET ingang wordt intern gesynchroniseerd op de processorklok. Bij het initialiseren mag de HOOG-naar-LAAG overgang van RESET niet eerder dan 50μs na power-up plaatsvinden om de 8088 de gelegenheid te geven volledig geïnitialiseerd te worden.

Indien INTR eerder dan negen klokcyclussen na het einde van RESET wordt gegeven, kan het voorkomen dat de processor een instructie uitvoert voordat hij reageert op de interruptie.

Alle 3-state uitgangen komen in de hoogimpedante toestand gedurende RESET, terwijl status gedurende de eerste klok na RESET aktief is en daarna hoog-impedant wordt.

Interrupt handelingen
De interrupt-operaties kunnen worden veroorzaakt door software of door hardware. De interrupties die het gevolg zijn van software en de software-aspecten van hardware-interrupties worden gespecificeerd in de beschrijving van de instructieset van het MCS-86 gebruikers-handboek. Hardware interrupties kunnen worden geklassificeerd als wel- of niet-maskeerbaar.

Na een interruptie wordt de besturing naar een nieuwe programma-lokatie overgebracht. Voor dit doel is op de absolute adressen tussen 0 en 3FFH een uit 256 elementen bestaande tabel gereserveerd die adrespointers bevat die verwijzen naar de interrupt service program lokaties (figuur 7/2.4-9). Elk element in de tabel is vier bytes groot en komt overeen met een interruptie-'type'. Een interrumperend apparaat levert tijdens het bevestigen van de interruptie (interrupt acknowledge) een 8-bit typenummer dat wordt gebruikt om via het juiste element naar de nieuwe lokatie voor het uitvoeren van de interruptieroutine te gaan.

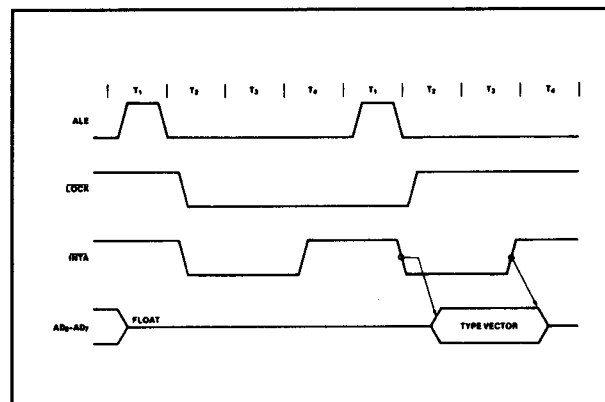
Niet-maskeerbare interruptie (NMI)

De processor heeft een enkele pen voor niet-maskeerbare interrupties (NMI) die een hogere prioriteit heeft dan de maskeerbare interrupt-request pen (INTR). Een typisch gebruik hiervan is bijvoorbeeld het aktiveren van een routine bij het wegvallen van de voeding. De NMI wordt getriggerd op de flank van een LAAG-naar-HOOG overgang. Aktivering van deze pen veroorzaakt een type 2 interruptie.

NMI moet langer dan twee klokcyclussen HOOG zijn maar behoeft niet te worden gesynchroniseerd op de klok. Elke volgende HOOG-overgang van NMI wordt op de chip gelatcht en wordt aan het einde van de actuele instructie bediend of tussen twee hele verplaatsingen (2 bytes in het geval van woordverplaatsingen) van een bloktype instructie.
Worst-case responsie op NMI treedt op bij vermenigvuldigen, delen en variabele verschuif-instructies. Er is geen voorschrift betreffende de neergaande flank; deze mag voor, tijdens of na de bediening van NMI komen. Een volgende hooggaande flank triggert een volgende reactie indien deze optreedt na het begin van de NMI-procedure. Het signaal mag in het algemeen geen logische spikes bevatten en moet vrij zijn van natrillingen op de neergaande flank.

Maskeerbare interrupties (INTR)

De 8088 heeft één interrupt request ingang (INTR) die intern met software gemaskeerd kan worden door het interrupt enable vlag-statusbit te resetten. Het interrupt request signaal is niveau-getriggerd en wordt intern gesynchroniseerd op de opgaande flank van elke CLK-cyclus. Om beantwoord te worden moet INTR aanwezig (HOOG) zijn gedurende de klokperiode voorafgaande aan het einde van de lopende instructie of het einde van een gehele verplaatsing (move) voor een blok-type instructie. Tijdens de behandeling van de interrupt routine worden verdere interrupties geblokkeerd. Het enable bit wordt gereset als deel van de reactie op een interruptie (INTR, NMI, software interrupt of single step) hoewel het vlagregister, dat automatisch op de stack wordt gezet, de toestand van de processor voorafgaande aan de interruptie weergeeft. Totdat het oude vlagregister is teruggebracht blijft het enable bit nul, tenzij het door een opdracht speciaal wordt geset.



Figuur 7/2.4-14: Bevestiging van een interruptie (interrupt acknowledge sequence).

Bij de behandeling van een interruptie (zie figuur 7/2.4-14) voert de processor twee opeenvolgende bevestigingscyclussen (interrupt acknowledge) uit. De 8088 zendt het LOCK signaal uit (alleen in de maxium mode) van T2 van de eerste buscyclus tot T2 van de tweede. Een lokaal bus 'hold' request wordt niet gehonoreerd tot het einde van de tweede buscyclus. In de tweede buscyclus wordt een byte opgehaald van het externe interruptie-systeem (bijvoorbeeld een 8259A - programmable interrupt controller - PIC) die de herkomst van de interruptie identificeert. Deze byte wordt met vier vermenigvuldigd om als pointer te worden gebruikt in de interrupt-vector opzoektabel. Een INTR signaal dat HOOG wordt gelaten zal voortdurend worden beantwoord binnen de beperkingen van de enable bit- en sample-periode. De interrupt return opdracht bevat een 'vlagpop' waarmee de status van het originele interrupt enable bit wordt teruggezet.

2.4 8088

HALT
Wanneer een software halt-instructie wordt uitgevoerd, geeft de processor aan dat hij in de halt-toestand gaat. Dit kan op twee manieren, afhankelijk van de gekozen mode. In de minimum mode levert de processor ALE na een vertraging van een klokcyclus om het systeem de gelegenheid te geven de halt-status te latchen. Halt-status is verkrijgbaar op IO/M, DT/R een SSO. In de maximum mode geeft de processor de juiste halt-status op S2, S1 en S0, terwijl de 8288 buscontroller een ALE levert. De 8088 zal de halt-toestand niet verlaten als tijdens halt een lokale bus-hold binnenkomt. In dat geval geeft de processor op het einde van de lokale bus-hold opnieuw de halt-indicatie. Met een interrupt request of een RESET wordt de 8088 uit de halt-toestand gedwongen.

Read/Modify/Write (semafoor) bewerkingen via LOCK
De processor geeft LOCK statusinformatie wanneer tijdens het uitvoeren van een opdracht direct aansluitende buscyclussen nodig zijn. De processor is hierdoor in staat om read/modify/write handelingen in het geheugen te verrichten (via de 'verwissel register met geheugen' instructie) zonder dat een andere systeem busmaster interveniërende geheugencyclussen ontvangt. Dit is nuttig om in multi-processorsystemen 'test en set lock' operaties mogelijk te maken. Het LOCK signaal wordt geaktiveerd (LAAG gemaakt) in de klokcyclus volgende op die waarin de software LOCK prefix instructie wordt gedecodeerd door de EU. Op het einde van de laatste buscyclus van de instructie die op de LOCK prefix instructie volgt, wordt het LOCK signaal gedeaktiveerd. Terwijl LOCK aktief is, worden alle interrupties gemaskeerd en een request op een RQ/GT-pen wordt vastgelegd om te worden gehonoreerd aan het einde van de LOCK.

Externe synchronisatie via TEST
Als een alternatief op de interrupties en algemene I/O mogelijkheden heeft de 8088 een TEST ingang die met software getest kan worden. Te allen tijde kan het programma een WAIT opdracht uitvoeren. Als in die tijd het TEST signaal inaktief (HOOG) is, wordt de uitvoering van het programma uitgesteld omdat de processor wacht tot TEST aktief wordt. Het moet tenminste vijf klokcyclussen aktief blijven. Tot die tijd wordt de WAIT instructie tlekens opnieuw uitgevoerd. Deze aktiviteit kost geen buscyclussen, aangezien de processor tijdens het wachten in een leegloop-toestand blijft. Als een bus HOLD binnenkomt gaan alle 8088 3-state drivers in de hoog-impedante toestand. Als interrupties enabled zijn, kunnen deze optreden terwijl de processor wacht. Gebeurt dit dan haalt de processor de WAIT instructie nog een keer extra op, voert de interrupt routine uit en haalt daarna de WAIT opdracht nogmaals op om hem weer uit te voeren.

Principiële systeem timing
In de minimum mode is de MN/MX-pen aan Vcc gelegd en zendt de processor buscontrol signalen uit die compatibel zijn met de busstructuur van de 8085. In de maximum mode ligt de MN/MX-pen aan GND en zendt de processor gecodeerde status-informatie uit die de 8288 bus-controller gebruikt om MULTIBUS compatibele busbesturingssignalen te genereren.

Systeem timing - minimum systeem
De leescyclus begint op T1 (zie figuur 7/2.4-13) met het aanbrengen van het adres latch enable (ALE) signaal. De neergaande achterflank van dit signaal wordt gebruikt om de adres-informatie die beschikbaar is op de adres/databus (AD0 t/m AD7) in de 8282/8283 te latchen. De adreslijnen A8 t/m A15 behoeven niet te worden gelatcht omdat deze gedurende de hele buscyclus geldig blijven. Van T1 tot T4 laat het IO/M-signaal een geheugen- of I/O-handeling zien. Bij T2 wordt het adres van de adres/databus verwijderd en gaat de bus in de hoog-impedante toestand. Het read control signaal verschijnt ook op T2. Door het leessig-

2.4 8088

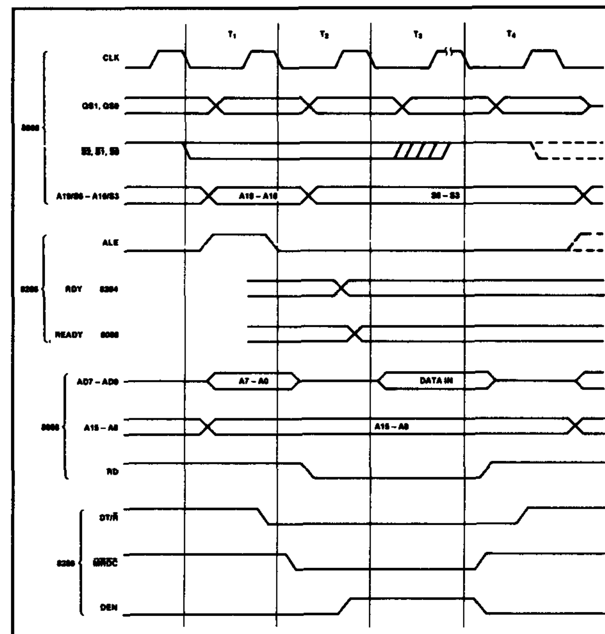
signaal ($\overline{RD}$) maakt de geadresseerde schakeling zijn databus drivers aktief voor de lokale bus. Enige tijd later zal geldige data op de bus beschikbaar zijn en zal de geadresseerde schakeling de READY-lijn HOOG zetten. Als de processor het leessignaal weer HOOG maakt, zet de geadresseerde schakeling zijn busdrivers weer in de hoogimpedante toestand. Indien een transceiver (8286/8287) nodig is om de lokale 8088 bus te bufferen, levert de 8088 de signalen DT/$\overline{R}$ en $\overline{DEN}$.

Een schrijfcyclus begint ook met het aanbrengen van ALE en de emissie van het adres. Het IO/$\overline{M}$-signaal wordt weer verstrekt om een geheugen- of I/O-handeling aan te geven. Op T2, onmiddellijk na het verschijnen van het adres, stuurt de processor de data naar de geadresseerde plaats. Deze data blijft minstens geldig tot het midden van T4. Gedurende T2, T3 en Tw geeft de processor het write control signaal. Het schrijfsignaal ($\overline{WR}$) wordt bij het begin van T2 aktief (in tegenstelling tot read, dat iets in T2 is vertraagd om de bus tijd te geven om te zweven).

Het belangrijkste verschil tussen de interrupt acknowledge cyclus en een leescyclus is dat het interrupt acknowledge signaal ($\overline{INTA}$) wordt aangebracht in plaats van het read signaal ($\overline{RD}$) en dat de adresbus zweeft (zie figuur 7/2.4-14). In de laatste van twee aaneensluitende $\overline{INTA}$-cyclussen wordt een byte informatie van de databus gelezen die door de interrupt systeemlogika (bijvoorbeeld een 8259A PIC) wordt geleverd. Deze byte identificeert de bron (type) van de interruptie. Hij wordt met vier vermenigvuldigd en gebruikt als pointer in de interrupt vector opzoektabel, zoals al eerder werd beschreven.

Bustiming - gemiddeld complexe systemen

Voor systemen met een gemiddelde complexiteit wordt de MN/$\overline{MX}$-pen aan GND gelegd en wordt het systeem uitgebreid met de 8288 buscontroller, een 8282/8283 latch voor het latchen van het systeemadres en een 8286/8287 transceiver voor busmanipulaties die boven de mogelijkheden van de 8088 uitgaan (zie figuur 7/2.4-15). De signalen ALE, $\overline{DEN}$ en DT/$\overline{R}$ worden nu door de 8288 gegenereerd in plaats van door de processor, hoewel hun timing vrijwel gelijk blijft. De status-uitgangen $\overline{S2}$, $\overline{S1}$ en $\overline{S0}$ van de 8088 leveren cyclustype informatie en worden 8288 ingangen. Deze buscyclus informatie specificeert read (code, data of I/O), write (data of I/O), interrupt acknowledge of software halt.



Figuur 7/2.4-15: Timing van complexe systemen.

De 8288 levert dan besturingssignalen die geheugen read of write, I/O read of write of interrupt acknowledge specificeren. De 8288 geeft, naar behoefte, twee typen write strobe af: normaal of geavanceerd. De normale write strobes hebben geldige data op de voorflank van write. De geavanceerde strobes hebben dezelfde timing als read strobes, waardoor data niet geldig is op de voorflank van write. De 8286/8287 transceiver ontvangt de gebruikelijke T en $\overline{OE}$ ingangssignalen van de DT/$\overline{R}$ en $\overline{DEN}$ uitgangen van de 8288.

2.4 8088

De pointer naar de interrupt vectortabel die tijdens de tweede $\overline{INTA}$ cyclus komt, kan afkomstig zijn van een 8259A op de lokale bus of de systeembus. Als de master 8259A priority interrupt controller op de lokale bus is aangesloten, is een TTL-poort nodig om de 8286/8287 transceiver te blokkeren wanneer tijdens de interrupt acknowledge en software 'poll' uit de master 8259A wordt gelezen.

De instructie-set

Overzicht

Zie de samenvatting van de instructieset (tabel 7/2.4-5) en de toelichtingen hierop (tabellen 7/2.4-3, 7/2.4-6 en 7/2.4-7).

Data Transfer **MOV = Move:**	76543210	76543216	76543210	76543210
Register / memory to / from register	100010dw	mod reg r/m		
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w=1
Immediate to register	1011w reg	data	data if w=1	
Memory to accumulator	1010000w	addr-low	addr-high	
Accumulator to memory	1010001w	addr-low	addr-high	
Register/memory to segment register	10001110	mod 0 reg r/m		
Segment register to register/memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/memory	11111111	mod 110 r/m		
Register	01010 reg			
Segment register	000 reg 110			
POP = Pop:				
Register/memory	10001111	mod 000 r/m		
Register	01011 reg			
Segment register	000 reg 111			
XCHG = Exchange:				
Register/memory with register	1000011w	mod reg r/m		
Register with accumulator	10010 reg			
IN = Input from:				
Fixed port	1110010w	port		
Variable port	1110110w			

OUT = Output to:	76543210	76543210	76543210	76543210
Fixed port	1110011w	port		
Variable port	1110111w			
XLAT = Translate byte to AL	11010111			
LEA = Load EA to register	10001101	mod reg r/m		
LDS = Load pointer to DS	11000101	mod reg r/m		
LES = *Load pointer to ES*	11000100	mod reg r/m		
LAHF = Load AH with flags	10011111			
SAHF = Store AH into flags	10011110			
PUSHF = Push flags	10011100			
POPF = Pop flags	10011101			
Arithmetic **ADD = Add:**				
Reg./memory with register to either	000000dw	mod reg r/m		
Immediate to register/memory	100000sw	mod 000 r/m	data	data if s: w=01
Immediate to accumulator	0000010w	data	data if w=1	
ADC = Add with carry:				
Reg./memory with register to either	000100dw	mod reg r/m		
Immediate to register/memory	100000sw	mod 010 r/m	data	data if s:w=01
Immediate to accumulator	0001010w	data	data if w=1	
INC = Increment:				
Register/memory	1111111w	mod 000 r/m		
Register	01000 reg			
AAA = ASCII adjust for add	00110111			
DAA = Decimal adjust for add	00100111			
SUB = Subtract:				
Reg./memory and register to either	001010dw	mod reg r/m		
Immediate from register/memory	100000sw	mod 101 r/m	data	data if s:w=01
Immediate from accumulator	0010110w	data	data if w=1	
SBB = Subtract with borrow:				
Reg./memory and register to either	000110dw	mod reg r/m		
Immediate from register/memory	100000sw	mod 011 r/m	data	data if s:w=01
Immediate from accumulator	0001110w	data	data if w=1	
DEC = Decrement:				
Register/memory	1111111w	mod 001 r/m		
Register	01001 reg			
NEG = Change sign	1111011w	mod 011 r/m		

CMP = Compare:	76543210	76543210	76543210	76543210
Register/memory and register	001110dw	mod reg r/m		
Immediate with register/memory	100000sw	mod 111 r/m	data	data if s:w=01
Immediate with accumulator	0011110w	data	data if w=1	
AAS = ASCII adjust for subtract	00111111			
DAS = Decimal adjust for subtract	00101111			
MUL = Multiply (unsigned)	1111011w	mod 100 r/m		
IMUL = Integer multiply (signed)	1111011w	mod 101 r/m		
AAM = ASCII adjust for multiply	11010100	00001010		
DIV = Divide (unsigned)	1111011w	mod 110 r/m		
IDIV = Integer divide (signed)	1111011w	mod 111 r/m		
AAD = ASCII adjust for divide	11010101	00001010		
CBW = *Convert byte to word*	10011000			
CWD = Convert word to double word	10011001			

2.4 8088

Logic

NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift logical/arithmetic left	110100vw	mod 100 r/m		
SHR = Shift logical right	110100vw	mod 101 r/m		
SAR = Shift arithmetic right	110100vw	mod 111 r/m		
ROL = Rotate left	110100vw	mod 000 r/m		
ROR = Rotate right	110100vw	mod 001 r/m		
RCL = Rotate through carry flag left	110100vw	mod 010 r/m		
RCR = Rotate through carry flag right	110100vw	mod 011 r/m		

AND = And:

Reg./memory and register to either	001000dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 100 r/m	data	data if w=1
Immediate to accumulator	0010010w	data	data if w=1	

TEST = And function to flags, no result:

Register/memory and register	1000010w	mod reg r/m		
Immediate data and register/memory	1111011w	mod 000 r/m	data	data if w=1
Immediate data and accumulator	1010100w	data	data if w=1	

OR = Or:

Reg./memory and register to either	000010dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 001 r/m	data	data if w=1
Immediate to accumulator	0000110w	data	data if w=1	

XOR = Exclusive Or:

Reg./memory and register to either	001100dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 110 r/m	data	data if w=1
Immediate to accumulator	0011010w	data	data if w=1	

String Manipulation	76543210	76543210	76543210
REP = Repeat	1111001z		
MOVS = Move byte/word	1010010w		
CMPS = Compare byte/word	1010011w		
SCAS = Scan byte/word	1010111w		
LODS = Load byte/word to AL/AX	1010110w		
STOS = Store byte/word from AL/A	1010101w		

Control Transfer

CALL = Call:

Direct within segment	11101000	disp-low	disp-high
Indirect within segment	11111111	mod 010 r/m	
Direct intersegment	10011010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 011 r/m	

JMP = Unconditional jump:

Direct within segment	11101001	disp-low	disp-high
Direct within segment short	11101011	disp	
Indirect within segment	11111111	mod 100 r/m	
Direct intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 101 r/m	

RET = Return from CALL:

Within segment	11000011		
Within seg. adding immediate to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment adding immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on equal/zero	01110100	disp	
JL/JNGE = Jump on less/not greater or equal	01111100	disp	
JLE/JNG = Jump on less or equal/not greater	01111110	disp	
JB/JNAE = Jump on below/not above or equal	01110010	disp	
JBE/JNA = Jump on below or equal/not above	01110110	disp	
JP/JPE = Jump on parity/parity even	01111010	disp	
JO = Jump on overflow	01110000	disp	
JS = Jump on sign	01111000	disp	
JNE/JNZ = Jump on not equal/not zero	01110101	disp	
JNL/JGE = Jump on not less/greater or equal	01111101	disp	
JNLE/JG = Jump on not less or equal/greater	01111111	disp	
JNB/JAE = Jump on not below/above or equal	01110011	disp	
JNBE/JA = Jump on not below or equal/above	01110111	disp	
JNP/JPO = Jump on not parity/parity odd	01111011	disp	
JNO = Jump on not overflow	01110001	disp	
JNS = Jump on not sign	01111001	disp	
LOOP = Loop CX times	11100010	disp	
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp	
JCXZ = Jump on CX zero	11100011	disp	

INT = Interrupt	76543210	76543210
Type specified	11001101	type
Type 3	11001100	
INTO = Interrupt on overflow	11001110	
IRET = Interrupt return	11001111	

Processor Control

CLC = Clear carry	11111000	
CMC = Complement carry	11110101	
STC = Set carry	11111001	
CLD = Clear direction	11111100	
STD = Set direction	11111101	
CLI = Clear interrupt	11111010	
STI = Set interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to external device)	11011xxx	mod xxx r/m
LOCK = Bus lock prefix	11110000	

Footnotes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if s:w = 01 then 16-bits of immediate data from the operand
it s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparsion with ZF FLAG

Segment Override Prefix

001 reg 110

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp high is absent
if mod = 10 then DISP = disp-high: disp low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) +DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

* except if mod = 00 and r/m = 110 then EA = disp-high:disp-low.

REG is assigned according to the following table

16-bit (w=1)	8-bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instruction which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

Tabel 7/2.4-5: Samenvatting van de instructieset van de 8088 processor.

2.4 8088

"reg" Field Bit Assignments:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

"mod" Field Bit Assignments:

mod xxx r/m

mod	Displacement
00	DISP = 0*, disp-low and disp-high are absent
01	DISP = disp-low sign-extended to 16-bits, disp-high is absent
10	DISP = disp-high: disp-low
11	r/m is treated as a "reg" field

"r/m" Field Bit Assignments:

r/m	Operand Address
000	(BX) + (SI) + DISP
001	(BX) + (DI) + DISP
010	(BP) + (SI) + DISP
011	(BP) + (DI) + DISP
100	(SI) + DISP
101	(DI) + DISP
110	(BP) + DISP
111	(BX) + DISP

DISP follows 2nd byte of instruction (before data if required).
*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

Tabel 7/2.4-3: Toelichting op de instructieset van de 8088 (fields).

IDENTIFIER	USED IN	EXPLANATION
destination	data transfer, bit manipulation	A register or memory location that may contain data operated on by the instruction, and which receives (is replaced by) the result of the operation.
source	data transfer, arithmetic, bit manipulation	A register, memory location or immediate value that is used in the operation, but is not altered by the instruction.
source-table	XLAT	Name of memory translation table addressed by *register* BX.
target	JMP, CALL	A label to which control is to be transferred directly, or a register or memory location whose *content* is the address of the location to which control is to be transferred indirectly.
short-label	cond. transfer, iteration control	A label to which control is to be conditionally transferred; must lie within −128 to +127 bytes of the first byte of the next instruction.
accumulator	IN, OUT	Register AX for word transfers, AL for bytes.
port	IN, OUT	An I/O port number; specified as an immediate value of 0-255, or register DX (which contains port number in range 0-64k).
source-string	string ops.	Name of a string in memory that is addressed by register SI; used only to identify string as byte or word and specify segment override, if any. This string is used in the operation, but is not altered.
dest-string	string ops.	Name of string in memory that is addressed by register DI; used only to identify string as byte or word. This string receives (is replaced by) the result of the operation.
count	shifts, rotates	Specifies number of bits to shift or rotate; written as immediate value 1 or register CL (which contains the count in the range 0-255).
interrupt-type	INT	Immediate value of 0-255 identifying interrupt pointer number.
optional-pop-value	RET	Number of bytes (0-64k, ordinarily an even number) to discard from stack.
external-opcode	ESC	Immediate value (0-63) that is encoded in the instruction for use by an external processor.
above-below	conditional jumps	Above and below refer to the relationship of two unsigned values.
greater-less	conditional jumps	Greater and less refer to the relationship of two signed values.

Tabel 7/2.4-6 Toelichting op de instructieset (identifiers).

Key to Operand Types

IDENTIFIER	EXPLANATION
(no operands)	No operands are written
register	An 8- or 16-bit general register
reg 16	An 16-bit general register
seg-reg	A segment register
accumulator	Register AX or AL
immediate	A constant in the range 0-FFFFH
immed8	A constant in the range 0-FFH
memory	An 8- or 16-bit memory location[1]
mem8	An 8-bit memory location[1]
mem16	A 16-bit memory location[1]
source-table	Name of 256-byte translate table
source-string	Name of string addressed by register SI
dest-string	Name of string, addressed by register DI
DX	Register DX
short-label	A label within −128 to +127 bytes of the end of the instruction
near-label	A label in current code segment
far-label	A label in another code segment
near-proc	A procedure in current code segment
far-proc	A procedure in another code segment
memptr16	A word containing the offset of the location in the current code segment to which control is to be transferred[1]
memptr32	A doubleword containing the offset and the segment base address of the location in another code segment to which control is to be transferred[1]
regptr16	A 16-bit general register containing the offset of the location in the current code segment to which control is to be transferred
repeat	A string instruction repeat prefix

(1) Any addressing mode — direct, register indirect, based, indexed, or based indexed — may be used

Tabel 7/2.4-7: Toelichting op de instructieset (operand typen).

2.4 8088

Adresseer modes
De instructieset van de 8088 maakt het adresseren van operands op verschillende manieren mogelijk. Bewerkingen met twee operands staan meestal gebruik van geheugen of een register als één operand toe, terwijl een register of een constante binnen de opdracht als de andere operand kan dienen. In het algemeen kunnen operands in geheugen **direct** worden geadresseerd met een 16-bit offset adres, of **indirect** met **basis** (BX of BP) en/of **index** (SI of DI)-registers toegevoegd aan een optionele 8- of 16-bit **displacement constante.** Het resultaat van een twee-operand bewerking kan naar elk van de twee bron-operands worden gestuurd, natuurlijk met uitzondering van in-line immediate constanten. Voor bewerkingen met enkele operands kunnen alle operands worden gebruikt met uitzondering van immeditate constanten. Bijna alle bewerkingen van de 8088 kunnen met 8- of 16-bit operands werken.

Geheugen operands, dit wil zeggen operands die in het geheugen verblijven, kunnen op vier manieren worden geadresseerd:
- direct via 16-bit offset adres;
- indirect via een basisregister, optioneel met een 8- of 16-bit verplaatsing (displacement);
- indirect via een indexregister, optioneel met een 8- of 16-bit displacement;
- indirect via de som van een basisregister en een indexregister, optioneel met een 8- of 16-bit displacement.

Zowel het dataregister BX als het pointerregister BP kan dienen als basisregister. Wanneer BX de basis is, bevindt de operand zich systeemgekozen in het actuele datasegment. Is BP de basis dan staat de operand systeemgekozen in het actuele stacksegment en wordt het SS segmentregister gebruikt om het echte adres van de operand te berekenen. Wanneer zowel basis- als indexregisters worden gebruikt, dan bevindt de operand zich systeemgekozen in het segment dat door het basisregister wordt bepaald. Wordt alleen een indexregister gebruikt dan staat de operand systeemgekozen in het actuele data-segment. Operands in geheugen kunnen, net als register- en immediate operands, 8- of 16-bits lang zijn.

De plaats van een operand in een 8088 register of in geheugen wordt gespecificeerd met maximaal drie velden bij elke instructie (zie tabel 7/2.4-3). Deze velden zijn het mode veld (mod field), het register veld (reg field) en het register/geheugen veld (r/m field).
Wanneer zij gebruikt worden, bezetten zij de tweede byte van de instructie-volgorde. Het mod-veld komt op de twee belangrijkste bits van de byte en specificeert hoe het r/m veld wordt gebruikt bij het lokaliseren van de operand. Het register dat in het r/m-veld wordt gespecificeerd kan òf de operand bevatten òf naar de plaats van de operand in het geheugen wijzen. Het reg-veld bezet de drie bits naast het mod-veld en bepaalt of een operand in een 8- of een 16-bit register staat.

Segment Override Prefixes
Wanneer het BP-register bij de berekening van het adres voor een geheugen-operand betrokken is, wordt het SS-segmentregister gebruikt om het fysieke adres te berekenen. Het fysieke adres van de meeste andere operands wordt met behulp van het DS-segmentregister berekend. In plaats van deze systeemgekozen segmentregisters kunnen andere worden gebruikt door de verwijzende instructie vooraf te laten gaan door een 'segment override prefix'.

Register Operands
De vier 16-bit dataregisters en de vier 16-bit pointerregisters kunnen, onderling verwisselbaar, dienen als operands in bijna alle 16-bit bewerkingen. Belangrijke uitzonderingen zijn vermenigvuldigen, delen, I/O en sommige string-handelingen die impliciet gebruik maken van het AX-register. De acht 8-bit registers van de HL-groep kunnen, ook onderling verwisselbaar, worden gebruikt in 8-bit handelingen.

2.4 8088

Vermenigvuldigen, delen en sommige stringbewerkingen maken impliciet gebruik van AL.

Immediate Operands
Bij alle twee-operand bewerkingen behalve vermenigvuldigen, delen en string-handelingen, mag een bron-operand binnen de instructie verschijnen als immediate data. Een 16-bit immediate operand die een hoogste byte heeft die alleen het teken van de laatste byte bevat, mag worden afgekort tot 8 bits.

Timing van de Instructieset
Tabel 7/2.4-4 laat zien hoeveel klokcyclussen de 8088 nodig heeft voor het uitvoeren van elke instructie. De 8088 heeft voor het ophalen of wegschrijven van 16-bit operands vier klokcyclussen meer nodig dan de 8086, aangezien hij met een 8-bit databus werkt. Bovendien moet men bedenken dat als de wachtrij leeg is, de uitvoering van de opdrachten tot minimaal vier klokcyclussen per instructie-byte wordt beperkt. Wanneer een subroutine bijvoorbeeld begint met een aantal kortdurende instructies zoals move immediates, register moves of pointer updates, zal de executietijd langer zijn dan in het handboek wordt gespecificeerd.

EA COMPONENTS		CLOCKS*
Displacement Only		6
Base or Index Only	(BX,BP,SI,DI)	5
Displacement + Base or Index	(BX,BP,SI,DI)	9
Base + Index	BP + DI, BX + SI	7
	BP + SI, BX + DI	8
Displacement + Base + Index	BP + DI + DISP BX + SI + DISP	11
	BP + SI + DISP BX + DI + DISP	12

*Add 2 clocks for segment override

Tabel 7/2.4-4: Aantal klokcyclussen voor het uitvoeren van de opdrachten.

Data transfer
Er kunnen vier klassen van data-overdracht bewerkingen worden onderscheiden: algemene, specifiek op de accumulator betrekking hebbende, adresobject overdrachten en vlag-overdrachten. Geen enkele overdracht-instructie heeft invloed op vlagposities, behalve SAHF en POPF.

Er staan vier algemene overdracht bewerkingen ter beschikking. Deze kunnen voor de meeste operands worden gebruikt, hoewel er bepaalde uitzonderingen zijn. De algemene overdrachten (behalve XCHG) zijn de enige bewerkingen waarbij een segmentregister als operand is toegestaan:
MOV: Move (verplaatsen);
PUSH: (pushen: het toevoegen van een element aan een wachtrij of stapelgeheugen);
POP: Pop (poppen: het behandelen van een element uit een wachtrij of stapelgeheugen);
XCHG: Exchange (verwisselen).

Er zijn drie adres-object overdracht bewerkingen:
LEA: Load Effective Address;
LDS: Load Pointer into DS;
LES: Load Pointer into ES.

Rekenkundige bewerkingen
De 8088 voert vier wiskundige basishandelingen uit op een aantal verschillende manieren. Er kunnen zowel 8- als 16-bit bewerkingen met of zonder teken worden uitgevoerd. Voor waarden met teken wordt de standaard two's complement schrijfwijze gebruikt. Optellen en aftrekken kunnen zowel met als zonder teken worden uitgevoerd, waarbij het eventuele gebruik van tekens blijkt uit gezette vlaggen. Om rekenkundige bewerkingen direct mogelijk te maken voor 'ongepakte decimale cijfers' of 'gepakte decimale representaties' kunnen correctie-handelingen worden uitgevoerd.

2.4 8088

Vlaggen
Om bepaalde eigenschappen van het resultaat van een bewerking te laten zien, kunnen zes vlagbits worden geset of gecleared. Meestal worden de volgende regels gevolgd.

- CF wordt geset als de handeling resulteert in een carry (bij een optelling) uit, of een borrow (bij een aftrekking) naar het hoogste bit van het resultaat, in andere gevallen is CF gecleared.
- AF wordt geset als de bewerking resulteert in een carry uit of een borrow naar de laagste vier bits van het resultaat, anders is AF gecleared.
- ZF wordt geset als het resultaat van een bewerking nul is en blijft anders gecleared.
- SF wordt geset als het hoogste bit van het resultaat van een handeling HOOG wordt, in andere gevallen blijft SF gecleared.
- PF wordt geset als de modulo 2-som van de laagste 8 bits van het resultaat van de bewerking 0 is (even pariteit), anders wordt PF gecleared (oneven pariteit).
- OF wordt geset als de bewerking een carry naar het hoogste bit van het resultaat tot gevolg heeft, maar geen carry uit het hoogste bit, of omgekeerd, anders blijft OF gecleared.

Optellen en aftrekken
Er zijn vijf soorten optelling mogelijk:

ADD:	Add (optellen);
ADC:	Add met carry;
INC:	Increment;
AAA:	Ongepakte BCD (ASCII) aanpassing voor optelling;
DAA:	Decimale aanpassing voor optelling.

Zeven manieren van aftrekken:

SUB:	Subtract (aftrekken);
SBB:	Subtract met borrow;
DEC:	Decrement;
NEG:	Negate (ontkennen: logische NIET);
CMP:	Compare (vergelijken);
AAS:	Ongepakte BCD (ASCII) aanpassing voor aftrekking;
DAS:	Decimale aanpassing voor aftrekking.

Vermenigvuldigen en delen
Er kunnen drie vermenigvuldigingen en drie delingen worden gedaan plus twee handelingen met teken-uitbreiding die het delen met teken ondersteunen:

MUL:	Multiply (vermenigvuldigen)
IMUL:	Integer multiply (vermenigvuldigen van gehele getallen);
AAM:	Ongepakte BCD (ASCII) aanpassing voor vermenigvuldigen;
DIV:	Divide (delen);
IDIV:	Integer divide (delen met gehele getallen);
AAD:	Ongepakte BCD (ASCII) aanpassing voor delen;
CBW:	Convert Byte to Word (zet byte om in woord);
CWD:	Convert Word to Double Word (maak dubbele woordlengte).

Logische enkel-operand instructies
Er staan negen logische enkele-operand instructies ter beschikking:

NOT:	Not (een's complement);
SHL:	Shift Left (naar links schuiven);
SAL:	Shift Arithmetic Left (rekenkundig naar links);
SHR:	Shift Right (naar rechts schuiven);
SAR:	Shift Arithmetic Right (rekenkundig naar rechts);
ROL:	Rotate Left (linksom wentelen);
ROR:	Rotate Right;
RCL:	Rotate through Carry Left (linksom door carry);
RCR:	Rotate through Carry Right.

String manipulatie
De 8088 heeft een groep 1-byte instructies die verschillende primitieve manipulaties van byte- of woord-strings uitvoeren. Deze primitieve handelingen kunnen herhaald worden uitgevoerd door de instructie te laten vooraf-

gaan door een speciale prefix. De enkele operatie-vormen mogen worden gecombineerd om complexe string-manipulaties te verrichten die worden herhaald door speciale iteratie-opdrachten.

Hardware Operation Control
Alle primitieve string-handelingen gebruiken het SI-register om de bron-operands te adresseren, waarvan wordt aangenomen dat zij zich in het actuele datasegment bevinden. Het DI-register wordt gebruikt om de bestemmings-operands te adresseren, waarvan wordt aangenomen dat zij in het actuele extra-segment staan. Als de DF-vlag wordt gecleared, worden de operand-pointers na elke bewerking verhoogd, eenmaal voor byte-handelingen en tweemaal voor woord-operaties. Als de DI-vlag wordt geset, worden de operand-pointers na elke operatie verlaagd.

Alle primitieve string-operatie opdrachten mogen worden voorafgegaan door een 1 byte prefix die aangeeft dat de handeling moet worden herhaald totdat de handelingenteller in CX tot nul is verlaagd. Voordat de handeling wordt herhaald wordt eerst getest of CX nul is.
De herhalings prefix-byte vormt ook een waarde die met de ZF-vlag wordt vergeleken. Als de primitieve handeling er een is die de ZF-vlag beïnvloedt en de ZF-vlag na een uitvoering van de primitieve handeling niet gelijk is aan de gevormde waarde, dan wordt de herhaling gestopt. Hierdoor kan het aftasten (scan) worden gebruikt als een scan-while of een scan-until.

Gedurende de uitvoering van een herhaalde primitieve operatie worden de operand-pointerregisters (SI en DI) en het operatietelregister (CX) na elke herhaling bijgewerkt, terwijl de instructie-pointer het offset-adres van het herhaal-prefixbyte vasthoudt (als dit tenminste direct voor de string-behandelingsopdracht staat).
Zodoende zal een onderbroken herhaalde handeling na de interruptie correct worden geïnterpreteerd.

Software Operation Control
De repeat prefix maakt snelle iteratie in een hardware herhaalde stringbehandeling mogelijk. De iteratie control-handelingen maken ook besturing van complexe string-handelingen door software loops mogelijk. Bij deze iteratie-handelingen worden, net als bij de repeat prefix, de operatie-teller bijgewerkt, getest of de handeling klaar is en de ZF-vlag getest.

Door de primitieve string-operaties en de iteratie control-handelingen met andere bewerkingen te combineren is het mogelijk uitgekiende en toch efficiënte string-manipulaties op te zetten. Een instructie die in deze context buitengewoon nuttig is, is XLAT; hiermee kan een byte die uit de ene string wordt gehaald, worden vertaald voordat hij in een tweede string wordt opgeborgen, of op een andere wijze bewerkt. De vertaling wordt uitgevoerd door de waarde in het AL-register te gebruiken als index in een tabel die door het RX-register wordt aangewezen. De vertaalde waarde die uit de tabel wordt gehaald vervangt dan de oorspronkelijke waarde in het AL-register.

Er zijn vijf primitieve string-operaties mogelijk, die alle zowel een byte instructie als een woord-instructie hebben:

MOVS:	Move (verplaats) byte of woord;
CMPS:	Compare (vergelijk) byte of woord;
SCAS:	Scan byte of woord (aftasten);
LODS:	Load byte of woord (laden);
STOS:	Store byte of woord (opbergen).

Control Transfer
Er kunnen vier klassen van besturingsoverdracht (control transfer) worden onderscheiden: calls, jumps en returns; conditional transfers; iteration control; interrupts.
Alle bewerkingen met besturingsoverdracht

2.4 8088

hebben tot gevolg (sommige conditioneel) dat de uitvoering van het programma op een nieuwe plaats in het geheugen begint, eventueel in een nieuw codesegment.

Calls, Jumps, Returns
Er zijn twee principiëel verschillende oproepen, sprongen en terugkeringen mogelijk: die met overdrachtsbesturing binnen het actuele codesegment en die met overdrachtsbesturing naar een willekeurig codesegment dat dan het actuele codesegment wordt. Er worden zowel directe als indirecte overdrachten ondersteund, de indirecte maken gebruik van de standaard adresseringsmodes. Intrasegment directe oproepen en sprongen (calls, respectievelijk jumps) specificeren een directe verplaatsing ten opzichte van zichzelf, waardoor positie-onafhankelijke codering mogelijk is. Voor overdrachten binnen ±128 bytes van de opdracht is een verkorte jump mogelijk die minder code gebruikt.

Conditional Transfers
De conditionele besturingsoverdrachten maken een sprong volgens verschillende Boole functies van het vlagregister. De bestemming moet binnen ±128 bytes van de opdracht liggen.

Iteration Control
De overdrachten voor herhalingsbesturingen vormen voor- en achterlopende beslissingslussen. De bestemming van de iteratiebesturingsoverdrachten moet ook binnen ±128 bytes van de opdracht liggen. Deze handelingen (vier) zijn vooral nuttig in samenwerking met de string-manipulaties:
LOOP: programmalus;
LOOPZ: loop while zero;
LOOPE: loop while equal;
LOOPNZ: loop while not zero;
LOOPNE: loop while not equal;
JCXZ: jump on CX zero.

Interrupts
De besturing van het uitvoeren van een programma kan worden overgedragen door middel van bewerkingen die hetzelfde effect hebben als externe interrupties. Alle onderbrekingen voeren een overdracht uit door de vlagregisters op de stack te zetten (net als bij PUSHF) en daarna indirect via een element een interrupt-transfer vector op te halen die in de absolute lokaties 0 tot en met 3FFH staat. Deze vector bevat een 4-bit element voor elk van de maximaal 256 verschillende soorten interrupties. Er zijn drie interruptie-overdrachtshandelingen:
INT: Interrupt;
INTO: Interrupt on Overflow;
IRET: Interrupt Return.

Single Step
Wanneer de TF-vlag is geset, genereert de processor een type 1 interruptie na het uitvoeren van elke opdracht. Tijdens de onderbreking wordt de TF-vlag na de 'push flags'-stap gecleared. Er zijn geen instructies voor een directe beïnvloeding van TF. In plaats daarvan moet de kopie van het vlagregister die door een eerdere interruptie op de stack werd gezet, worden gemodificeerd zodat de volgende 'interrupt return' de gesette TF terugbrengt. Dit maakt dat het stap-voor-stap doorlopen van een routine een diagnostische taak kan krijgen.
Als de stap-voor-stap uitgevoerde opdracht zelf de TF-vlag cleart, zal de type 1 interruptie op het einde van deze opdracht toch nog optreden. Als de stap-voor-stap opdracht een interruptie opwekt of als een vrijgegeven externe interruptie optreedt voordat de stap-voor-stap instructie is beëindigd, zal de type 1 interrupt-afhandeling plaatsvinden na de interruptvolgorde van de opgewekte of externe interruptie maar voordat de eerste opdracht van die interrupt-serviceroutine wordt uitgevoerd.

Definities en functies van de aansluitpennen

De hierna volgende beschrijving van de functies van de aansluitpennen gelden voor

2.4 8088

de 8088 in zowel de minimum- als de maximum-mode. De hierin genoemde 'lokale bus' is de direct gemultiplexte interface-verbinding met de 8088 (zonder rekening te houden met externe bus-buffers).

AD7 - AD0: Adres/Databus
(Input/Output, 3-state)
Deze lijnen vormen de tijd-gemultiplexte geheugen-/IO-adresbus (T1) en databus (T2, T3, Tw en T4). De lijnen zijn aktief-HOOG en komen gedurende interrupt acknowledge en lokale bus 'hold acknowledge' in de hoog-impedante toestand.

A15 - A8: Adresbus
(Output, 3-state)
Op deze lijnen staan gedurende de hele buscyclus (T1 - T4) de adresbits 8 t/m 15, zodat ze niet door ALE behoeven te worden gelatcht. A15 - A8 zijn aktief-HOOG en zijn gedurende de hierboven genoemde acknowledge-tijden hoog-impedant.

A19/S6, A18/S5, A17/S4, A16/S3: Adres/Status
(Output, 3-state)
Gedurende T1 zijn dit de vier belangrijkste adreslijnen voor geheugen-operaties. Tijdens I/O-handelingen zijn deze lijnen LAAG. Bij geheugen- en I/O-handelingen is statusinformatie hierop beschikbaar tijdens T2, T3, Tw en T4. S6 is altijd LAAG. De status van het interrupt enable-vlagbit (S5) wordt aan het begin van elke klokcyclus bijgewerkt. S4 en S3 worden volgens tabel 7/2.4-8 gecodeerd. Deze informatie geeft aan welk segmentregister op dat moment wordt gebruikt om toegang tot de data te krijgen. Tijdens lokale bus 'hold acknowledge' zijn deze lijnen hoog-impedant.

$\overline{RD}$: Read
(Output, 3-state)
De read-strobe geeft aan dat de processor bezig is met een geheugen- of I/O-leescyclus, afhankelijk van de toestand van de IO/$\overline{M}$-pen of $\overline{S2}$. Het signaal wordt gebruikt om schakelingen die met de lokale 8088-bus zijn verbonden in te lezen. $\overline{RD}$ is aktief-LAAG tijdens T2, T3 en Tw van elke leescyclus en blijft gegarandeerd HOOG in T2 totdat de 8088-bus heeft gezweefd. De uitgang is hoog-impedant tijdens 'hold acknowledge'.

S4	S3	Characteristics
0	0	Alternate Data
0	1	Stack
1	0	Code or None
1	1	Data

Tabel 7/2.4-8: Codering van het gebruikte segment-register door middel van S4 en S3.

READY: Ready
(Input)
READY is de bevestiging van het geadresseerde geheugen- of I/O-apparaat dat het de data-overdracht zal afmaken. Het RDY signaal van geheugen of I/O wordt door de 8284 clock generator gesynchroniseerd om zo READY te vormen. Het signaal is aktief-HOOG.

INTR: Interrupt Request
(Input)
Interrupt request is een flankgetriggerde ingang die gedurende de laatste klokcyclus van elke instructie wordt afgetast om te bepalen of de processor een interrupt acknowledge-handeling moet uitvoeren. Via een vector-opzoektabel die in het geheugen staat wordt naar een subroutine gesprongen. De ingang kan intern worden gemaskeerd door het interrupt enable-bit met software te resetten. INTR wordt intern gesynchroniseerd en is aktief-HOOG.

$\overline{TEST}$: Test
(Input)
Deze ingang wordt bekeken door de 'wait for test' opdracht. Als de $\overline{TEST}$-ingang LAAG is,

2.4 8088

gaat de uitvoering door, anders wacht de processor in een 'stationaire' toestand. Deze ingang wordt intern tijdens elke klokcyclus gesynchroniseerd op de voorflank van CLK.

NMI: Non-Maskable Interrupt
(Input)
NMI is een flankgetriggerde ingang die een type 2 interruptie veroorzaakt. Via een vector-opzoektabel in het geheugen wordt naar een subroutine gesprongen. NMI kan niet intern worden gemaskeerd met software. Een LAAG-naar-HOOG overgang leidt de onderbreking in op het einde van de lopende instructie. Deze ingang wordt intern gesynchroniseerd.

RESET: Reset
(Input)
RESET laat de processor onmiddellijk zijn lopende aktiviteit beëindigen. Het signaal moet minstens vier klokcyclussen aktief-HOOG zijn en start de uitvoering opnieuw wanneer het weer LAAG wordt.
RESET wordt intern gesynchroniseerd.

CLK: Clock
(Input)
De klok verzorgt de basis-timing van de processor en de buscontroller. De klok is asymmetrisch met een aan/uit-verhouding (duty-cycle) van 33% om een optimale timing mogelijk te maken.

Vcc: Voedingsspanning
(+5V)
De voedingsspanning moet +5V ±10% bedragen.

GND: Aarde (OV)

MN/$\overline{MX}$: Minimum/Maximum
(Input)
Bepaalt de mode waarin de processor werkt. Beide modes worden hierna besproken.

Minimum mode pennen
De volgende functiebeschrijvingen gelden voor de 8088 minimum mode (als MN/$\overline{MX}$ = Vcc).

IO/M: Status
(Output, 3-state)
Deze statuslijn is een geïnverteerde maximum mode $\overline{S2}$ en wordt gebruikt om onderscheid te maken tussen een geheugen- en een I/O-toegang. IO/$\overline{M}$ wordt geldig tijdens de T4 die aan een buscyclus voorafgaat en blijft geldig tot de laatste T4 van de cyclus (I/O=HOOG, M=LAAG). IO/$\overline{M}$ is hoog-impedant tijdens een lokale bus 'hold acknowledge'.

$\overline{WR}$: Write
(Output, 3-state)
De write-strobe geeft aan dat de processor bezig is met een schrijfcyclus naar geheugen of I/O, afhankelijk van het IO/$\overline{M}$-signaal. $\overline{WR}$ is aktief tijdens T2, T3 en Tw van elke schrijfcyclus. Het is aktief-LAAG en hoog-impedant tijdens lokale bus 'hold acknowledge'.

$\overline{INTA}$: Interrupt Acknowledge
(Output, 3-state)
$\overline{INTA}$ wordt gebruikt als lees-strobe bij interrupt acknowledge-cyclussen. Het is aktief-LAAG tijdens T2, T3 en Tw van elke interrupt bevestigingscyclus. $\overline{INTA}$ is hoog-impedant tijdens 'hold acknowledge'.

ALE: Address Latch Enable
(Output)
Dit signaal wordt door de processor gegeven om het adres in de 8282/8283 adreslatch te plaatsen. Het is een HOOG-puls die aktief is gedurende de tijd dat de klok laag is bij T1 van elke klokcyclus.

DT/$\overline{R}$: Data Transmit/Receive
(Output, 3-state)
Dit signaal is nodig wanneer voor een minimum systeem een 8286/8287 databus transceiver gebruikt moet worden. Hiermee wordt

2.4 8088

de richting van de datastroom door de transceiver bepaald. Logisch is DT/$\overline{R}$ gelijkwaardig aan $\overline{S1}$ in de maximum mode en de timing ervan is dezelfde als voor IO/$\overline{M}$ (T=HOOG, R=LAAG). Dit signaal is hoog-impedant bij een lokale 'hold acknowledge'.

$\overline{DEN}$: Data Enable
(Output, 3-state)
Data enable werkt als output enable voor de 8286/8287 in een minimum systeem met een transceiver. $\overline{DEN}$ is aktief-LAAG tijdens de toegang tot geheugen en I/O en tijdens $\overline{INTA}$ cyclussen. Bij een lees- of $\overline{INTA}$-cyclus is het aktief vanaf het midden van T2 tot het midden van T4, terwijl het bij een schrijfcyclus aktief is vanaf het begin van T2 tot het midden van T4. $\overline{DEN}$ is hoog-impedant tijdens een lokale bus 'hold acknowledge'.

HOLD (Input), HLDA (Output): Hold
HOLD geeft aan dat een andere master een lokale bus 'hold' vraagt. Om te worden bevestigd, moet HOLD aktief-HOOG worden. De processor die het 'hold' verzoek ontvangt, maakt HLDA in het midden van T4 of T1 als antwoord HOOG. Tegelijk met de verandering op HLDA zal de processor de lokale bus en de controllijnen hoog-impedant maken. Nadat HOLD LAAG is bevonden, maakt de processor HLDA LAAG en indien de processor nog een cyclus moet uitvoeren zal hij opnieuw de lokale bus en controllijnen besturen.

$\overline{SS0}$: Status
(Output)
De statuslijn is logisch gelijkwaardig aan $\overline{S0}$ in de maximum mode. De combinatie van $\overline{SS0}$, IO/$\overline{M}$ en DT/$\overline{R}$ maakt het voor het systeem mogelijk de status van de lopende buscyclus compleet te decoderen (tabel 7/2.4-9).

Maximum mode pennen
De volgende functiebeschrijvingen gelden voor een 8088, 8288 systeem in de maximum mode (MN/$\overline{MX}$=GND).

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SS0}$	Characteristics
1	0	0	Interrupt Acknowledge
1	0	1	Read I/O Port
1	1	0	Write I/O Port
1	1	1	Halt
0	0	0	Code Access
0	0	1	Read Memory
0	1	0	Write Memory
0	1	1	Passive

Tabel 7/2.4-9: Complete decodering van de status van de lopende buscyclus.

$\overline{S2}$, $\overline{S1}$, $\overline{S0}$: Status
(Output, 3-state)
De codering van deze statuslijnen is te zien in tabel 7/2.4-10. De status is aktief als de klok HOOG is tijdens T4, T1 en T2 en wordt weer passief (1, 1, 1) tijdens T3 of Tw als READY HOOG is. De 8288 buscontroller gebruikt de status om alle benodigde geheugen- en I/O-besturingssignalen op te wekken.
Elke verandering van $\overline{S2}$, $\overline{S1}$ of $\overline{S0}$ tijdens T4 wordt gebruikt om het begin van een nieuwe klokcyclus aan te geven en de terugkeer naar de passieve toestand tijdens T3 of Tw betekent het einde van een buscyclus. Tijdens 'hold acknowledge' zijn deze uitgangen hoog-impedant. Gedurende de eerste klokcyclus nadat RESET aktief wordt, zijn deze signalen aktief-HOOG. Na deze eerste klok worden zij weer hoog-impedant.

$\overline{RQ}$/$\overline{GT0}$, $\overline{RQ}$/$\overline{GT1}$: Request/Grant
(Input/Output)
De request/grant pennen worden door andere lokale busmasters gebruikt om de processor te dwingen de lokale bus aan het einde van de lopende buscyclus vrij te geven. Elke pen is bidirectioneel, waarbij $\overline{RQ}$/$\overline{GT0}$ een hogere prioriteit heeft dan $\overline{RQ}$/$\overline{GT1}$.
$\overline{RQ}$/$\overline{GT}$ heeft een interne optrekweerstand en mag dus los blijven.

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Characteristics
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

Tabel 7/2.4-10: Codering van de statuslijnen S2, S1 en S0.

De request/grant volgorde is als volgt (zie ook figuur 7/2.4-13):

1 - Een puls (1 CLK breed) van een andere busmaster laat aan de 8088 zien dat er een lokale bus-request ('hold') is (puls 1).

2 - Gedurende de volgende T4 of T1 van de processor laat de 8088 met puls 2 aan de vragende master weten dat hij heeft toegestaan dat de bus zwevend wordt en dat hij bij de volgende CLK in de 'hold acknowledge' toestand gaat. De bus-interface van de processor wordt tijdens 'hold acknowledge' logisch losgekoppeld van de lokale bus.

3 - Een 1 CLK brede puls van de vragende master laat aan de 8088 weten (puls 3) dat het 'hold' verzoek bijna ten einde is en dat de 8088 de lokale bus bij de volgende CLK weer kan opeisen. De processor komt dan op T4. Elke master-master wisseling van de lokale bus levert drie pulsen op. Na elke wisseling moet een 'dode CLK' komen. De pulsen zijn aktief-LAAG.

$\overline{\text{LOCK}}$

(Output, 3-state)

De LOCK uitgang geeft aan dat andere busmasters de besturing van de systeembus niet mogen overnemen zolang $\overline{\text{LOCK}}$ aktief is (LAAG). Het $\overline{\text{LOCK}}$ signaal wordt geaktiveerd door de 'LOCK' prefix instructie en blijft aktief tot de volgende opdracht is beëindigd. Deze uitgang is hoog-impedant tijdens 'hold acknowledge'.

QS1, QS0: Queue Status

(Output)

QS1 en QS0 leveren status-informatie die extern volgen van de interne 8088 instructie wachtrij mogelijk maakt (zie tabel 7/2.4-11). De status van de wachtrij (queue) is geldig tijdens de CLK-cyclus waarna de queue-operatie wordt verricht.

QS1	QS0	Characteristics
0	0	No Operation
0	1	First Byte of Op Code from Queue
1	0	Empty the Queue
1	1	Subsequent Byte from Queue

Tabel 7/2.4-11: Statusinformatie van de wachtrij (queue) door middel van QS1 en QS0.

Elektrische eigenschappen

In tabel 7/2.4-12 zijn de gelijkstroom-eigenschappen van de 8088 te zien, terwijl tabel 7/2.4-13 met de bijbehorende figuren 7/2.4-16 en 7/2.4-17 (minimum mode) en tabel 7/2.4-14 met de figuren 7/2.4-18 en 7/2.4-19 (maximum mode) een indruk geven van het AC-gedrag tijdens bedrijf.

2.4 8088

D.C. CHARACTERISTICS

8088: $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.5$	V	
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.0$ mA
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = 400\ \mu A$
I_{CC}	Power Supply Current		275	mA	
I_{LI}	Input Leakage Current		±10	μA	$V_{IN} = V_{CC}$
I_{LO}	Output Leakage Current		±10	μA	$0.45V \leq V_{OUT} \leq V_{CC}$
V_{CL}	Clock Input Low Voltage	−0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.9	$V_{CC}+1.0$	V	
C_{IN}	Capacitance of Input Buffer (All input except AD_0-AD_7 RQ/GT)		10	pF	fc = 1 MHz
C_{IO}	Capacitance of I/O Buffer (AD_0-AD_7 RQ/GT)		20	pF	fc = 1 MHz

Tabel 7/2.4-12: Gelijkstroom-eigenschappen van de 8088.

2.4 8088

A.C. CHARACTERISTICS

8088: $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$

8088 MINIMUM COMPLEXITY SYSTEM TIMING REQUIREMENTS

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLCL	CLK Cycle Period	200	2000	ns	
TCLCH	CLK Low Time	$(\frac{2}{3}\text{TCLCL})-15$		ns	
TCHCL	CLK High Time	$(\frac{1}{3}\text{TCLCL})+2$		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		ns	
TCLDZ	Data In Hold Time	10		ns	
TR1VCL	RDY Setup Time into 8284 (See Notes 1, 2)	35		ns	
TCLR1X	RDY Hold Time into 8284 (See Notes 1, 2)	0		ns	
TRYHCH	READY Setup Time into 8088	$(\frac{2}{3}\text{TCLCL})-15$		ns	
TCHRYX	READY Hold Time into 8088	30		ns	
TRYLCL	READY Inactive to CLK (See Note 3)	-8		ns	
THVCH	Hold Setup Time	35		ns	
TINVCH	INTR, NMI, $\overline{\text{TEST}}$ Setup Time (See Note 2)	30		ns	

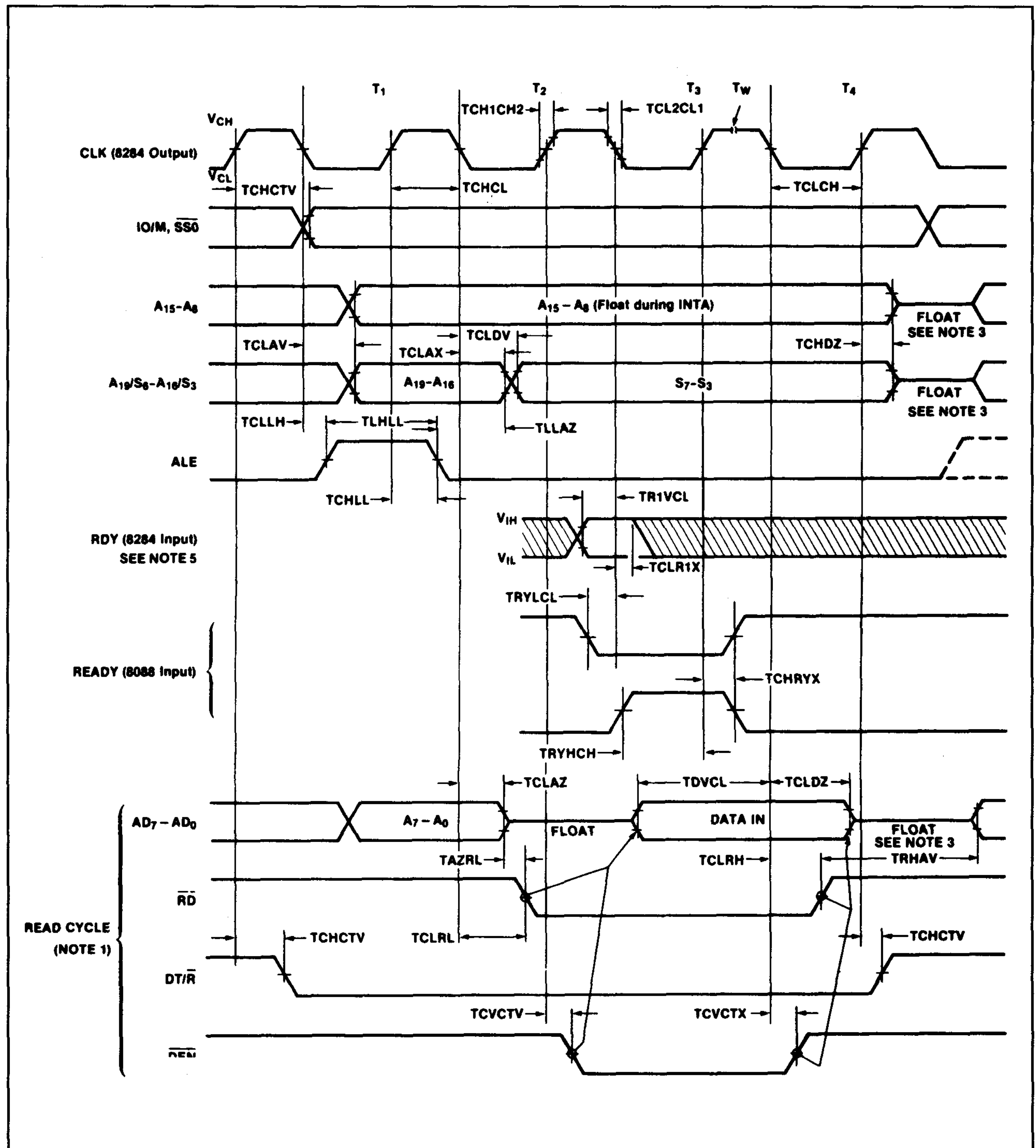
TIMING RESPONSES

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLAV	Address Valid Delay	15	110	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	80	ns	
TLHLL	ALE Width	TCLCH-20		ns	
TCLLH	ALE Active Delay		80	ns	
TCHLL	ALE Inactive Delay		85	ns	
TLLAZ	ALE Inactive to Address Float	TCHCL-10		ns	
TCLDV	Data Valid Delay	15	110	ns	C_L = 20-100 pF for all 8088 Outputs
TCHDZ	Data Float Delay	TCLAX	85	ns	
TWHDZ	Data Hold Time After $\overline{\text{WR}}$	TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	110	ns	
TCHCTV	Control Active Delay 2	15	110	ns	
TCVCTX	Control Inactive Delay	10	110	ns	
TAZRL	Address Float to READ Active	0		ns	
TCLRL	$\overline{\text{RD}}$ Active Delay	10	165	ns	
TCLRH	$\overline{\text{RD}}$ Inactive Delay	10	150	ns	
TRHAV	$\overline{\text{RD}}$ Inactive to Next Address Active	TCLCL-45		ns	
TCLHAV	HLDA Valid Delay	10	160	ns	

NOTES:
1. SIGNAL AT 8284 SHOWN FOR REFERENCE ONLY.
2. SETUP REQUIREMENT FOR ASYNCHRONOUS SIGNAL ONLY TO GUARANTEE RECOGNITION AT NEXT CLK.
3. APPLIES ONLY TO T2 STATE.

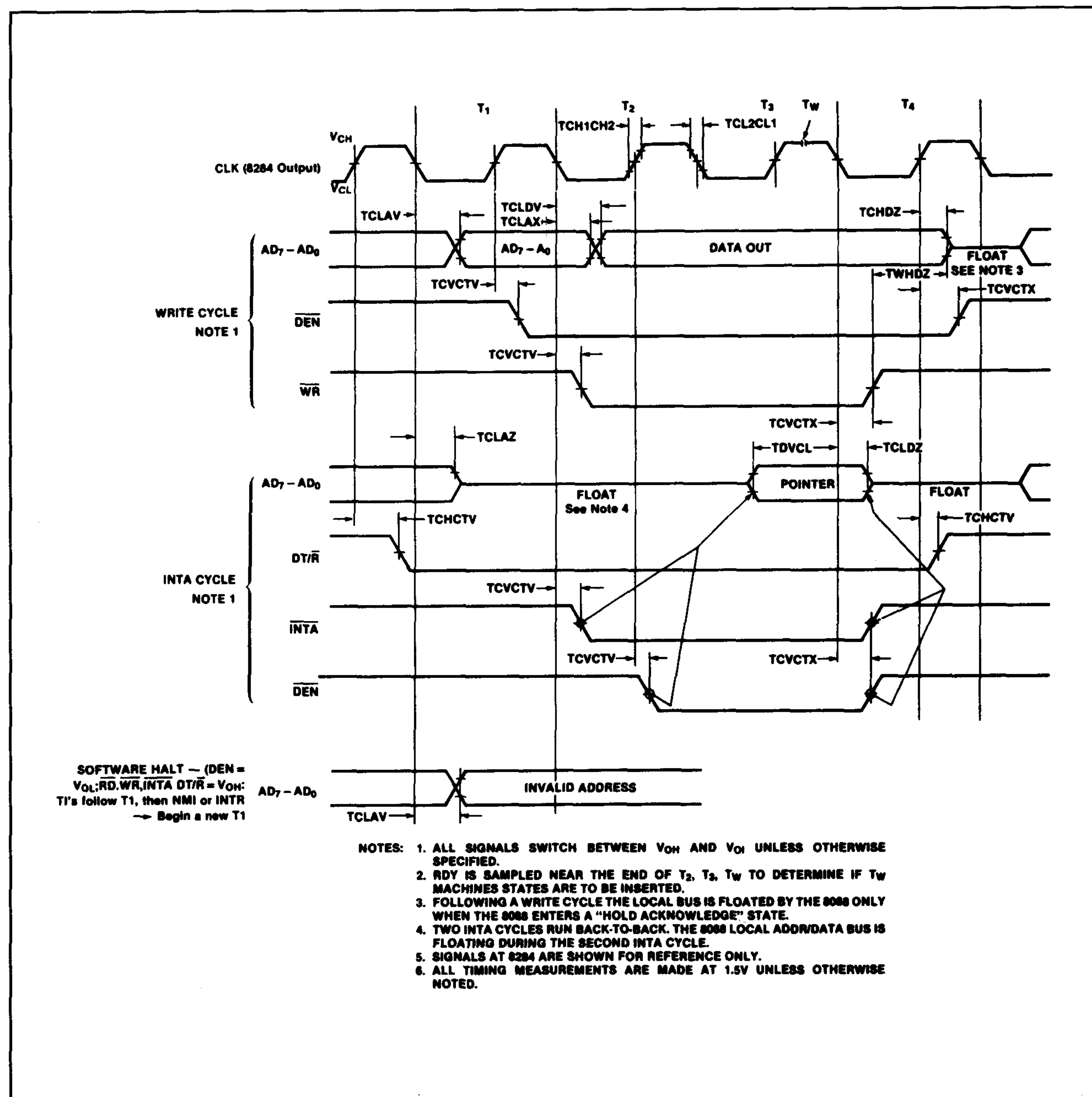
Tabel 7/2.4-13: Timing van de 8088 in de minimum mode.

2.4 8088



Figuur 7/2.4-16: Bustiming - minimum mode systeem.

2.4 8088



Figuur 7/2.4-17: Bustiming - minimum mode systeem (vervolg).

2.4 8088

8088 MAX MODE SYSTEM (USING 8288 BUS CONTROLLER)
TIMING REQUIREMENTS

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLCL	CLK Cycle Period	200	2000	ns	
TCLCH	CLK Low Time	(⅔TCLCL)-15		ns	
TCHCL	CLK High Time	(⅓TCLCL)+2		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		ns	
TCLDZ	Data In Hold Time	10		ns	
TR1VCL	RDY Setup Time into 8284 (See Notes 1, 2)	35		ns	
TCLR1X	RDY Hold Time into 8284 (See Notes 1, 2)	0		ns	
TRYHCH	READY Setup Time into 8088	(⅔TCLCL)-15		ns	
TCHRYX	READY Hold Time into 8088	30		ns	
TRYLCL	READY Inactive to CLK (See Note 4)	-8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{TEST}$) (See Note 2)	30		ns	
TGVCH	$\overline{RQ}/\overline{GT}$ Setup Time	30		ns	
TCHGX	$\overline{RQ}$ Hold Time into 8086	40		ns	

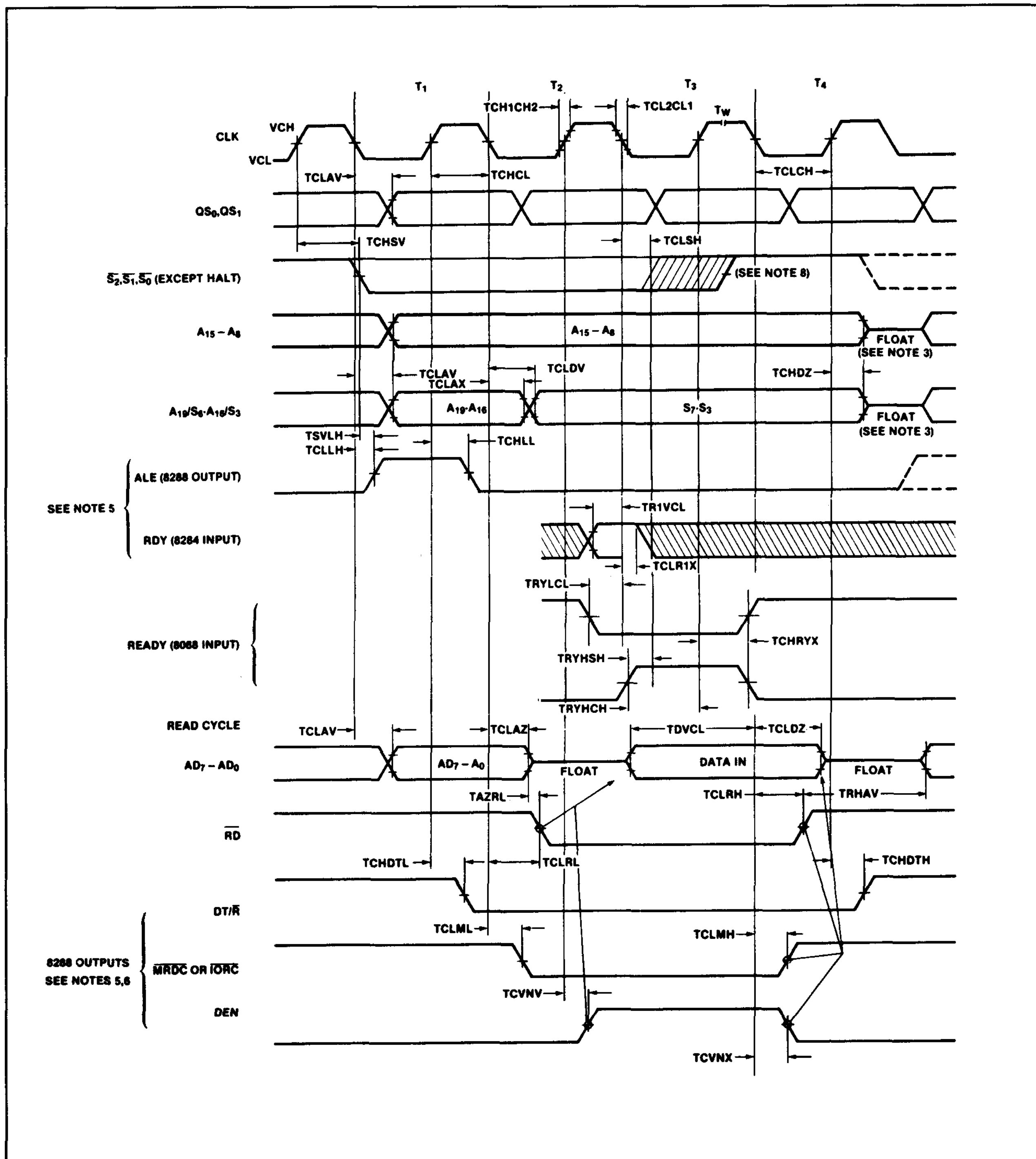
TIMING RESPONSES

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLML	Command Active Delay (See Note 1)	10	35	ns	
TCLMH	Command Inactive Delay (See Note 1)	10	35	ns	
TRYHSH	READY Active to Status Passive (See Note 3)		110	ns	
TCHSV	Status Active Delay	10	110	ns	
TCLSH	Status Inactive Delay	10	130	ns	
TCLAV	Address Valid Delay	15	110	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	80	ns	
TSVLH	Status Valid to ALE High (See Note 1)		15	ns	
TSVMCH	Status Valid to MCE High (See Note 1)		15	ns	
TCLLH	CLK Low to ALE Valid (See Note 1)		15	ns	
TCLMCH	CLK Low to MCE High (See Note 1)		15	ns	
TCHLL	ALE Inactive Delay (See Note 1)		15	ns	C_L = 20-100 pF for all 8088 Outputs
TCLMCL	MCE Inactive Delay (See Note 1)		15	ns	
TCLDV	Data Valid Delay	15	110	ns	
TCHDZ	Data Float Delay	TCLAX	85	ns	
TCVNV	Control Active Delay (See Note 1)	5	45	ns	
TCVNX	Control Inactive Delay (See Note 1)	10	45	ns	
TAZRL	Address Float to Read Active	0		ns	
TCLRL	RD Active Delay	10	165	ns	
TCLRH	RD Inactive Delay	10	150	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL-45		ns	
TCHDTL	Direction Control Active Delay (See Note 1)		50	ns	
TCHDTH	Direction Control Inactive Delay (See Note 1)		30	ns	
TCLGL	$\overline{GT}$ Active Delay		85	ns	
TCLGH	$\overline{GT}$ Inactive Delay		85	ns	

NOTES: 1. SIGNAL AT 8284 OR 8288 SHOWN FOR REFERENCE ONLY.
2. SETUP REQUIREMENT FOR ASYNCHRONOUS SIGNAL ONLY TO GUARANTEE RECOGNITION AT NEXT CLK.
3. APPLIES ONLY TO T3 AND WAIT STATES.
4. APPLIES ONLY TO T2 STATE.

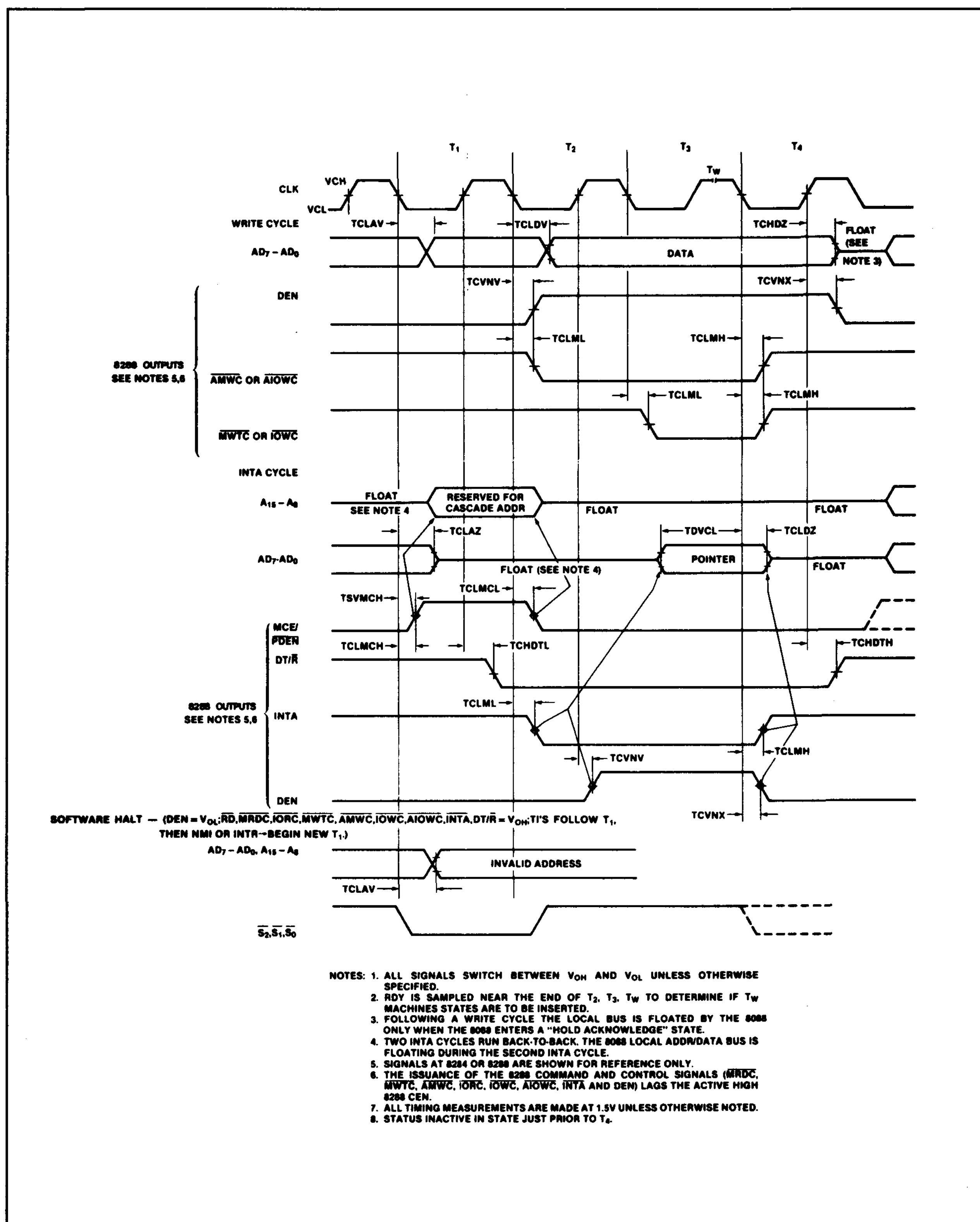
Tabel 7/2.4-14: Timing van de 8088 in de maximum mode.

2.4 8088



Figuur 7/2.4-18: Bustiming - maximum mode systeem (met gebruik van een 8288)..

2.4 8088



NOTES:
1. ALL SIGNALS SWITCH BETWEEN V_{OH} AND V_{OL} UNLESS OTHERWISE SPECIFIED.
2. RDY IS SAMPLED NEAR THE END OF T_2, T_3, T_W TO DETERMINE IF T_W MACHINES STATES ARE TO BE INSERTED.
3. FOLLOWING A WRITE CYCLE THE LOCAL BUS IS FLOATED BY THE 8088 ONLY WHEN THE 8088 ENTERS A "HOLD ACKNOWLEDGE" STATE.
4. TWO INTA CYCLES RUN BACK-TO-BACK. THE 8088 LOCAL ADDR/DATA BUS IS FLOATING DURING THE SECOND INTA CYCLE.
5. SIGNALS AT 8284 OR 8288 ARE SHOWN FOR REFERENCE ONLY.
6. THE ISSUANCE OF THE 8288 COMMAND AND CONTROL SIGNALS ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{AMWC}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{AIOWC}$, $\overline{INTA}$ AND DEN) LAGS THE ACTIVE HIGH 8288 CEN.
7. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.
8. STATUS INACTIVE IN STATE JUST PRIOR TO T_4.

Figuurl 7/2.4-19: Bustiming - maximum mode systeem (vervolg).

2.4 8088

7/2.5

Z80 (Z8400)

Inleiding

CP/M processor

De Z8400, zoals de CPU (Centrale Processing Unit) van de Z80-familie officieel heet, vormt de kern van de nog steeds veelgebruikte microcomputers die onder het zogenaamde CP/M (Control Program for Microcomputers) besturingsprogramma werken. De Z80 is de krachtigste en meest veelzijdige 8 bits microprocessor die behalve de complete broncode van de 8080A nog 80 instructies meer kent. Door de dubbele registerset zijn zeer snelle context-wisselingen en interrupt-afhandelingen mogelijk. Door de op de chip aanwezige programmeerbare refresh logika is het toepassen van dynamische geheugens zeer eenvoudig. Voor de vier traditionele functies van een microcomputersysteem (parallelle I/O, seriële I/O, tellen/timing en direct memory acces) wordt de Z80 CPU bijgestaan door enkele populaire familieleden: de Z80 PIO, Z80 SIO, Z80 DART, Z80 CTC en Z80 DMA. Voor deze computers is nog zeer veel programmatuur voorhanden.

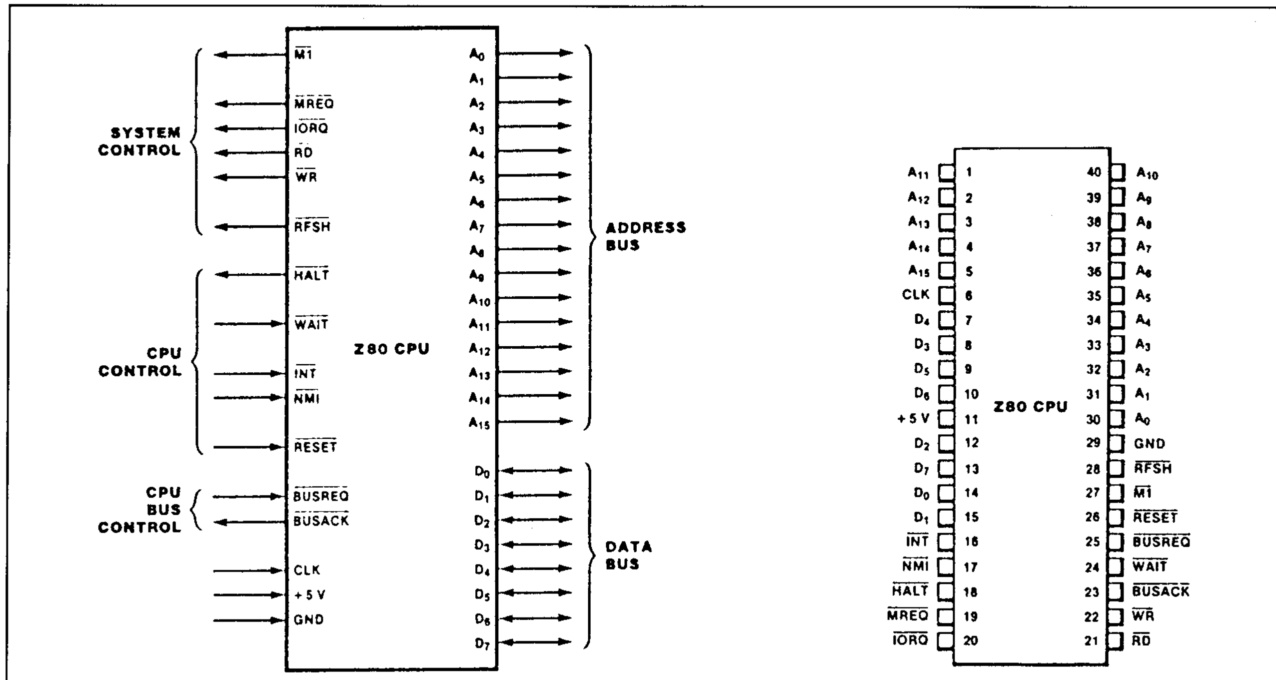
Algemene gegevens

- De instructieset bevat 158 instructies. De 78 instructies van de 8080A zijn opgenomen in een subset. Software voor de 8080A draait ook op de Z80.
- Alle in- en uitgangen TTL-compatibel.
- Verschillende kloksnelheden:
  6 MHz voor Z8400 B (Z80B);
  4 MHz voor Z8400 A (Z80A);
  2,5 MHz voor Z8400 (Z80).
- Aansluiting van dynamische of statische geheugens mogelijk met minimale logika.
- Dynamische RAM refresh-teller op de chip aanwezig.
- Enkelfase +5 V clock en enkele +5 V voeding.
- Leverbaar in NMOS en CMOS (Z84C00)
- String-, bit-, byte- en woord-operaties mogelijk door uitgebreide instructieset.
- Interrupt systeem eenvoudig in cascade schakelbaar.
- Interrupt afhandeling op 3 manieren mogelijk.
- 40-pens DIL-behuizing (zie figuur 7/2.5-1), CMOS ook in 44-pens PLCC.
- Normale (0 tot +70°C), industriële (−40 tot +85°C) en militaire (−55 tot +125°C) bedrijfstemperaturen.
- Leveranciers:
  Zilog: Z8400, Z8400A, Z8400B, Z8400/A/B
  SGS/ATES: Z8400, Z8400A, Z8400B, Z8400/A/B
  NEC: µPD780 (2,5 MHz), µPD780-1 (4 MHz)
  Mostek: MK3880, MK3880A, MK3880-4

Korte beschrijving

De interne registers van de Z80 CPU omvatten 208 bits lees/schrijf-geheugen die toegankelijk zijn voor de gebruiker. Hieronder vallen twee sets van zes registers voor algemene doeleinden, die apart als 8-bits registers of in paren als 16-bits registers kunnen worden gebruikt. Verder zijn er twee stellen accumulatoren en vlagregisters. Een groep 'exchange' instructies maakt beide sets hoofd- of alternatieve registers toegankelijk voor de programmeur. De alternatieve

2.5 Z80 (Z8400)



Figuur 7/2.5-1: Penfuncties en aansluitingen van de Z8400.

(om-en-om) set maakt werken in de 'voorgrond-achtergrond' mode mogelijk of kan worden gereserveerd voor zeer snelle interruptafhandelingen.
De Z80 heeft ook een Stack Pointer, een Program Counter, twee Index Registers, een Refresh Register (teller) en een Interrupt Register.
Opname van de Z80 CPU in een systeem is zeer eenvoudig, aangezien een enkele +5 V voeding nodig is en alle uitgangssignalen door volledige decodering en timing geschikt zijn voor standaard geheugen- en periferie schakelingen. In het blokschema (figuur 7/2.5-2) zijn de primaire functies van de Z80 processor te zien.

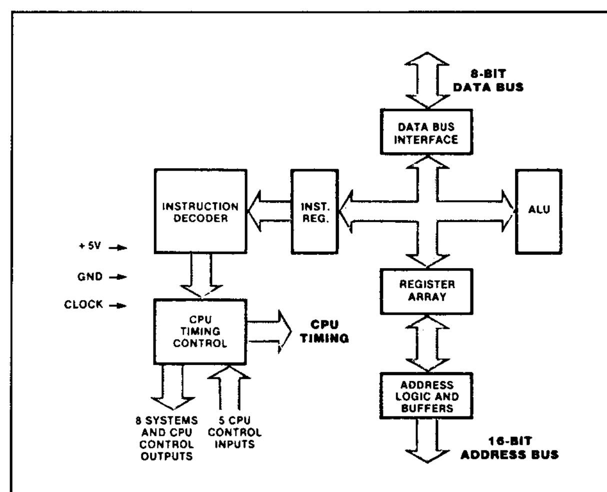
Definities en functies van de aansluitpennen

A_0–A_{15}: Adresbus

Deze aktief-HOGE 3-state uitgangen vormen een 16-bit adresbus, waarmee tot 64 kB geheugen- en I/O-data kan worden geadresseerd.

$\overline{\text{BUSACK}}$: Bus Acknowledge

Deze aktief-LAGE uitgang meldt aan het onderdeel dat om gebruik van de bus verzoekt, dat de adresbus, de databus en de besturingssignalen $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ en $\overline{\text{WR}}$ van



Figuur 7/2.5-2: Blokschema van de Z8400.

2.5 Z80 (Z8400)

de CPU in de hoog-impedante toestand zijn gegaan. Nu kan de externe schakeling deze lijnen besturen.

BUSREQ: Bus Request
Deze aktief-LAGE ingang heeft een hogere prioriteit dan NMI en wordt altijd gezien aan het einde van de lopende machine-cyclus. BUSREQ dwingt de adresbus, de databus en de besturingssignalen MREQ, IORQ, RD en WR van de CPU om in de hoog-impedante toestand te gaan zodat andere schakelingen de besturing van deze lijnen kunnen overnemen. BUSREQ wordt meestal als 'wired-OR' geschakeld, waarvoor dan een externe optrekweerstand nodig is.
Te lange BUSREQ perioden door uitvoerige DMA-handelingen gaan ten koste van een correcte refresh-functie voor de dynamische RAM's.

D_0–D_7: Databus
Deze aktief-HOGE, 3-state in/-uitgangen vormen een 8-bit bidirectionele databus die wordt gebruikt voor het uitwisselen van data tussen geheugen en I/O.

HALT: Halt state
De aktief-LAGE HALT-uitgang geeft aan dat de CPU een Halt-instructie heeft uitgevoerd en op een niet-maskeerbare interrupt of een maskeerbare interrupt (met enabled masker) wacht voordat de werking kan worden hervat. In deze toestand voert de CPU NOP's uit om geheugen-refresh te handhaven.

INT: Interrupt Request
Interrupt request-signalen voor deze aktief-LAGE ingang worden opgewekt door I/O-schakelingen. De CPU staat een verzoek om gebruik van de bus toe aan het einde van de lopende instructie, mits de interne, door software bestuurbare Interrupt Enable Flip-flop (IFF) is gezet. INT wordt meestal als 'wired-OR' gebruikt, waardoor een externe optrekweerstand nodig is.

IORQ: Input/Output Request
Deze aktief-LAGE, 3-state uitgang geeft aan dat op de laagste helft van de adresbus een geldig I/O-adres staat voor een I/O lees- of schrijfhandeling. IORQ wordt ook tegelijk met M1 gegenereerd tijdens een interrupt acknowledge cyclus om aan te geven dat een 'interrupt response vector' op de databus kan worden geplaatst.

M1: Machine Cycle One
Het M1-signaal op deze aktief-LAGE uitgang geeft samen met MREQ aan dat de lopende machine-cyclus de opcode fetch-cyclus van een instructie-executie is.
Wordt M1 gecombineerd met IORQ, dan wordt een interrupt acknowledge cyclus aangegeven.

MREQ: Memory Request
Deze aktief-LAGE, 3-state uitgang geeft aan dat op de adresbus een geldig adres voor een geheugenlees- of schrijfhandeling staat.

NMI: Non-Maskable Interrupt
Deze aktief-LAGE ingang heeft een hogere prioriteit dan INT. NMI wordt altijd erkend aan het einde van de lopende instructie (onafhankelijk van de status van de interrupt enable flip-flop) en dwingt de CPU automatisch opnieuw te starten op lokatie 0066H (hexadecimaal).

RD: Memory Read
Deze aktief-LAGE, 3-state uitgang geeft aan dat de CPU data uit het geheugen of uit een I/O-schakeling wil lezen. Dit signaal moet door het geadresseerde I/O-circuit of het adres worden gebruikt om data op de databus van de CPU te klokken.

RESET: Reset
Het RESET-signaal op deze aktief-LAGE ingang initialiseert de CPU als volgt: het reset de interrupt enable flip-flop, cleart de PC en de registers I en R en zet de interrupt status in mode 0. Tijdens het resetten gaan de adres- en databussen in de hoog-impedante

toestand en alle besturingssignalen naar de niet-aktieve toestand. Let op dat RESET minimaal drie klokcyclussen aktief moet zijn om de reset-handeling te volbrengen.

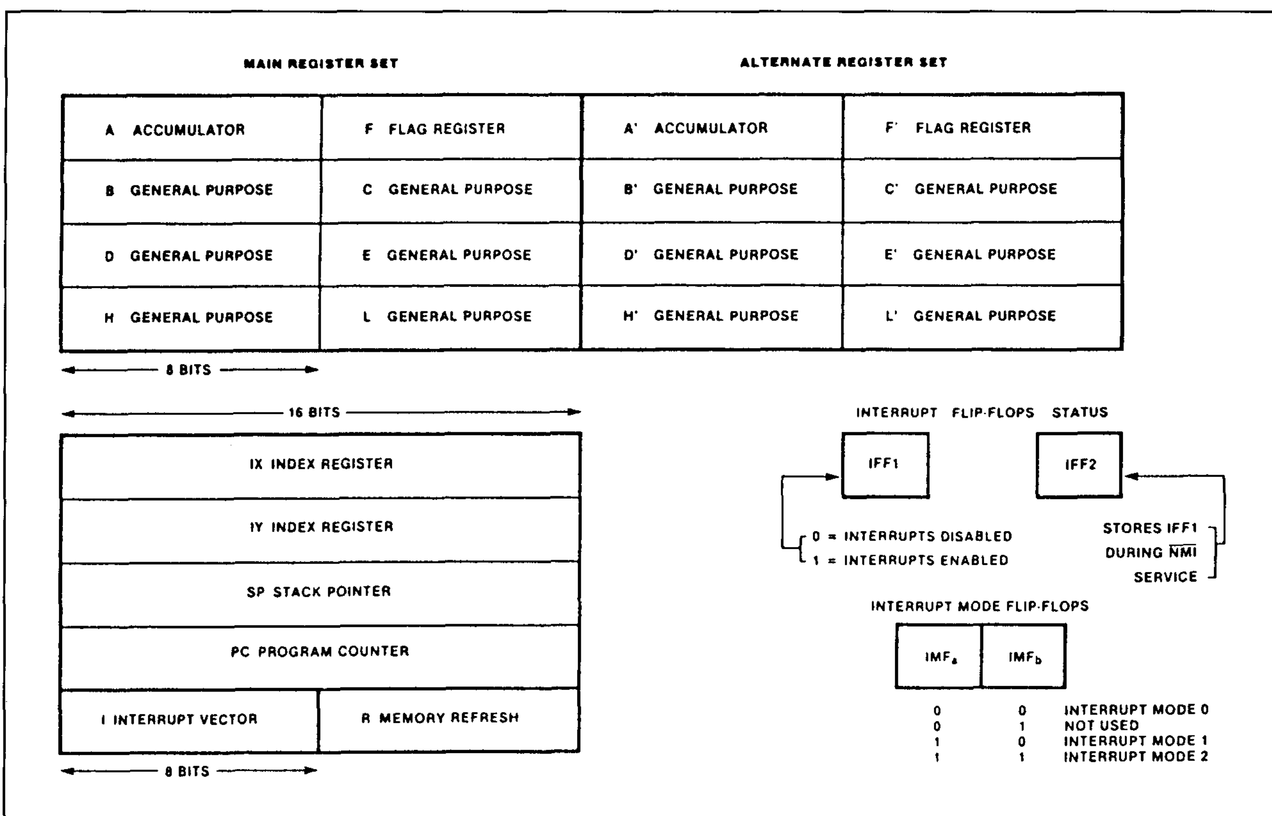
RFSH: Refresh
Het RFSH-signaal van deze aktief-LAGE uitgang geeft samen met MREQ aan dat de laagste 7-bits van de adresbus van het systeem kunnen worden gebruikt als refresh-adres voor de dynamische geheugens van het systeem.

WAIT: Wait
Deze aktief-LAGE ingang meldt aan de CPU dat het geadresseerde geheugen of I/O-circuit niet klaar is voor een data-overdracht. Zolang dit signaal aktief is, gaat de CPU in een Wait-state. Langdurige WAIT-perioden kunnen de CPU ervan weerhouden om het dynamisch geheugen correct te refreshen.

WR: Memory Write
Deze aktief-LAGE, 3-state uitgang geeft aan dat op de databus van de CPU geldige data staat die in de geadresseerde geheugen- of I/O-lokatie moet worden opgeborgen.

Registers

Overzicht
In figuur 7/2.5-3 zijn drie groepen registers van de Z80 CPU te zien.
De eerste groep bestaat uit dubbele sets 8-bits registers: een hoofdset en een alternatieve set (aangeduid met een ′, bijvoorbeeld A′). Beide sets bestaan uit het Accumulator Register, het Vlag (Flag) Register en zes registers voor algemene doeleinden (general-purpose). De overdracht van data tussen deze dubbele sets registers wordt geregeld door uitwisselings ('exchange') instructies. Als gevolg hiervan kan sneller worden ge-



Figuur 7/2.5-3: Interne registers.

reageerd op interrupties en is eenvoudige, efficiënte implementatie van veelzijdige programmeringstechnieken zoals achtergrond-voorgrond dataverwerking mogelijk.
De tweede groep registers bestaat uit zes registers met toegewezen functies. Het zijn de registers I (Interrupt), R (Refresh), IX en IY (Index) en de SP (Stack Pointer) en PC (Program Counter).
De derde groep heeft twee interrupt status flip-flops en een extra stel flip-flops die worden gebruikt bij de identificatie van de interrupt mode op elk willekeurig tijdstip.
In tabel 7/2.5-1 wordt meer informatie over deze registers gegeven.

Interrupties

$\overline{\text{NMI}}$ en $\overline{\text{INT}}$

De Z8400 accepteert twee interrupt-signalen: $\overline{\text{NMI}}$ en $\overline{\text{INT}}$. Het $\overline{\text{NMI}}$ is een niet-maskeerbare interruptie en heeft de hoogste prioriteit. $\overline{\text{INT}}$ is een interruptie met lagere prioriteit, aangezien voor de werking hiervan vereist is dat door middel van software op de interrupties wordt gereageerd (enable). Zowel $\overline{\text{NMI}}$ als $\overline{\text{INT}}$ kan met meerdere randapparaten worden verbonden in een zogenaamde 'wired-OR' configuratie. Zie ook figuur 7/2.5-24.

Niet-maskeerbare Interruptie ($\overline{\text{NMI}}$)

De niet-maskeerbare interruptie kan niet worden tegen gehouden door software en zal daardoor te allen tijde door de CPU worden geaccepteerd. $\overline{\text{NMI}}$ wordt meestal alleen gereserveerd voor behandeling van interrupties die de hoogste prioriteit hebben. Wanneer bijvoorbeeld een storing in de voeding wordt gedetecteerd, moet het lopende programma ordelijk worden afgesloten. Na herkenning van het $\overline{\text{NMI}}$-signaal springt de

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Stores addresses or data temporarily. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Tabel 7/2.5-1: Interne Registers van de Z8400.

CPU naar de herstart-lokatie 0066H (wanneer $\overline{BUSREQ}$ tenminste niet aktief is). Meestal bevat het programma dat op dit adres begint de interrupt service routine.

Maskeerbare interruptie ($\overline{INT}$)
Onafhankelijk van de door de gebruiker gekozen interrupt mode, volgt de reactie van de Z80 op een maskeerbare interruptie een gemeenschappelijke timing cyclus. Nadat de CPU een interruptie heeft gedetecteerd (vooropgesteld dat interrupties zijn toegestaan (enabled) en dat $\overline{BUSREQ}$ niet aktief is), begint een speciale interrupt-afhandelingscyclus. Dit is een speciale ophaalcyclus ($\overline{MI}$), waarin $\overline{IORQ}$ aktief wordt in plaats van $\overline{MREQ}$ zoals in een normale $\overline{MI}$-cyclus. Bovendien wordt deze speciale $\overline{MI}$-cyclus automatisch met twee $\overline{WAIT}$-toestanden verlengd om voldoende tijd te hebben voor het bevestigen van het interrupt-request en om de interrupt-vector op de bus te plaatsen.

Mode 0 Interruptie
Deze mode is compatibel met de interrupt-service procedures van de 8080 processor. Het onderbrekende apparaat zet een instructie op de databus die dan zesmaal wordt uitgevoerd door de CPU. Dit is meestal een herstart-instructie die een onvoorwaardelijke jump naar één van de acht mogelijke herstart-lokaties op pagina nul van het geheugen initialiseert.

Mode 1 Interruptie
De mode 1 operatie is bijna dezelfde als die voor $\overline{NMI}$. Het belangrijkste verschil is dat de mode 1 interruptie alleen het vector-adres 0038H heeft.

Mode 2 Interruptie
In deze interruptie-mode wordt zo efficiënt mogelijk gebruik gemaakt van de mogelijkheden van de Z80 microprocessor en zijn naaste familie. Het onderbrekende randapparaat kiest tijdens de interrupt acknowledge cyclus het startadres van de interrupt service-routine door een 8-bits adresvector op de databus te plaatsen. De hoogste 8-bits van het interrupt service-routine adres wordt geleverd door het I (Interrupt) register. Deze flexibiliteit bij het selecteren van het interrupt service-routine adres maakt dat het randapparaat van verschillende service-routines gebruik kan maken. Deze routines mogen op elke beschikbare plek in het geheugen worden gezet. Omdat het interrumperende apparaat de laagste 8-bits van de 2-bytes vector levert, moet bit 0 (A_0) een nul zijn.

Prioriteit bij de interrupties (Cascade-schakeling en geneste interrupties)
De interruptie-prioriteit van elk randapparaat wordt bepaald door zijn plaats in een cascade-schakeling. Elk apparaat in de keten heeft een interrupt enable ingang (IEI) en een interrupt enable uitgang (IEO) die met het volgende apparaat met lagere prioriteit is verbonden. De IEI-ingang van het eerste apparaat in de cascade-keten is met een HOOG niveau verbonden. Het eerste apparaat heeft de hoogste prioriteit, terwijl elk opeenvolgende apparaat een overeenkomstige lagere prioriteit heeft. Op deze wijze kan de CPU de interruptie met de hoogste prioriteit kiezen van verschillende onderbrekende randapparaten.
Het interrumperende apparaat blokkeert zijn IEO-lijn naar het volgende apparaat, totdat zijn interruptroutine is afgewerkt. Hierna wordt de IEO-lijn HOOG gemaakt, zodat de volgende apparaten met een lagere prioriteit bediend kunnen worden.
De Z80 CPU zal eventuele hangende interrupties of interrupties die binnenkomen terwijl hij met een interrupt service-routine bezig is 'nesten' (in een wachtrij plaatsen).

Interrupt Enable/Disable
De twee flip-flops IFF_1 en IFF_2 die ook in de beschrijving van de registers staan, worden gebruikt om de interrupt-status van de CPU aan te geven. De werking van de flip-flops wordt beschreven in tabel 7/2.5-2.

2.5 Z80 (Z8400)

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{INT}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{INT}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{INT}$ enabled
LD A,I instruction execution	•	•	IFF_2 → Parity flag
LD A,R instruction execution	•	•	IFF_2 → Parity flag
Accept $\overline{NMI}$	0	IFF_1	IFF_1 → IFF_2 (Maskable interrupt $\overline{INT}$ disabled)
RETN instruction execution	IFF_2	•	IFF_2 → IFF_1 at completion of an $\overline{NMI}$ service routine.

Tabel 7/2.5-2: Toestanden van de Flip-Flop's.

CPU Timing

Inleiding

De Z80 CPU voert de instructies uit door een bepaalde volgorde van handelingen te volgen:

- Lezen of schrijven naar geheugen;
- Lezen of schrijven naar I/O;
- Interrupt bevestigen.

De basis klokperiode wordt een T-tijd of cyclus genoemd, terwijl drie of meer T-cyclussen een 'machine-cyclus' wordt genoemd (bijvoorbeeld M1, M2 of M3). Machinecyclussen kunnen door de CPU worden verlengd wanneer die automatisch een of meer WAIT-toestanden inbrengt of door de gebruiker wanneer die WAIT-toestanden invoegt.
In tabel 7/2.5-3 zijn de bij de figuren 7/2.5-4 tot en met 7/2.5-11 behorende nummers en overeenkomstige schakeltijden te zien. Voor schakeltijden die afwijken van de in tabel 7/2.5-3 genoemde, kunnen de parameters volgens tabel 7/2.5-4 worden berekend.

Instructie Opcode Fetch

De CPU plaatst bij het begin van de cyclus de inhoud van de Program Counter (PC) op

Footnotes to AC Characteristics

Number	Symbol	Z80	Z80A	Z80B
1	TcC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC
2	TwCh	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed
7	TdA(MREQf)	TwCh + TfC − 75	TwCh + TfC − 65	TwCh + TfC − 50
10	TwMREQh	TwCh + TfC − 30	TwCh + TfC − 20	TwCh + TfC − 20
11	TwMREQl	TcC − 40	TcC − 30	TcC − 30
26	TdA(IORQf)	TcC − 80	TcC − 70	TcC − 55
29	TdD(WRf)	TcC − 210 .	TcC − 170	TcC − 140
31	TwWR	TcC − 40	TcC − 30	TcC − 30
33	TdD(WRf)	TwCl + TrC − 180	TwCl + TrC − 140	TwCl + TrC − 140
35	TdWRr(D)	TwCl + TrC − 80	TwCl + TrC − 70	TwCl + TrC − 55
45	TdCTr(A)	TwCl + TrC − 40	TwCl + TrC − 50	TwCl + TrC − 50
50	TdM1f(IORQf)	2TcC + TwCh + TfC − 80	2TcC + TwCh + TfC − 65	2TcC + TwCh + TfC − 50

AC Test Conditions:
V_{IH} = 2.0 V
V_{IL} = 0.8 V
V_{IHC} = V_{CC} -0.6 V
V_{ILC} = 0.45 V
V_{OH} = 2.0 V
V_{OL} = 0.8 V
FLOAT = ±0.5 V

Tabel 7/2.5-4: Berekening van afwijkende schakeltijden.

2.5 Z80 (Z8400)

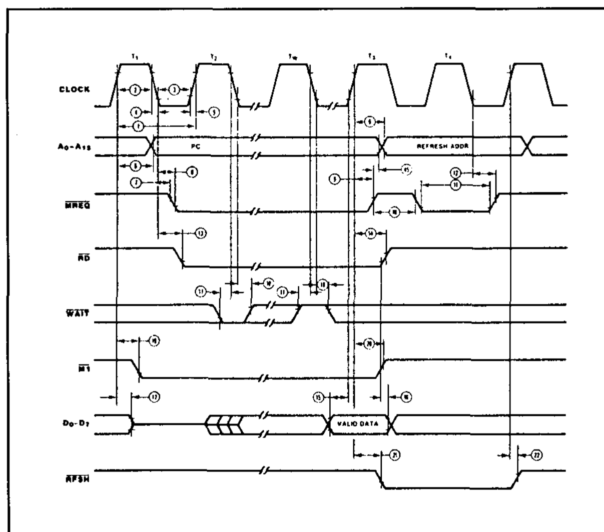
Number	Symbol	Parameter	Z80 CPU Min (ns)	Z80 CPU Max (ns)	Z80A CPU Min (ns)	Z80A CPU Max (ns)	Z80B CPU Min (ns)	Z80B CPU Max (ns)
1	TcC	Clock Cycle Time	400*		250*		165*	
2	TwCh	Clock Pulse Width (High)	180*		110*		65*	
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000
4	TfC	Clock Fall Time	—	30	—	30	—	20
5	TrC	Clock Rise Time		30		30		20
6	TdCr(A)	Clock ↑ to Address Valid Delay	—	145	—	110	—	90
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	125*	—	65*	—	35*	—
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	—	100	—	85	—	70
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	170*		110*		65*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	360*	—	220*	—	135*	—
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	—	130	—	95	—	80
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	—	100	—	85	—	70
15	TsD(Cr)	Data Setup Time to Clock ↑	50		35		30	
16	ThD(RDr)	Data Hold Time to $\overline{RD}$ ↑	—	0	—	0	—	0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	70	—	70	—	60	—
18	ThWAIT(Cf)	$\overline{WAIT}$ Hold Time after Clock ↓	—	0	—	0	—	0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	—	130	—	100	—	80
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay		130		100		80
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	—	180	—	130	—	110
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	—	150	—	120	—	100
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	—	110	—	85	—	70
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	—	100	—	85	—	70
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	60		50		40	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	320*	—	180*	—	110*	—
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	—	90	—	75	—	65
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	110	—	85	—	70
29	TdD(WRf)	Data Stable prior to WR ↓	190*		80*	—	25*	
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80		70
31	TwWR	$\overline{WR}$ Pulse Width	360*	—	220*	—	135*	—
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	—	100	—	80	—	70
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	20*	—	-10*	—	-55*	—
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	—	80	—	65	—	60
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	120*		60*		30*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	—	300	—	300	—	260
37	TwNMI	$\overline{NMI}$ Pulse Width	80	—	80	—	70	—
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	80	—	50	—	50	—
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0	—	0	—	0	—
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		120		100		90
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	—	110	—	100	—	90
42	TdCr(Dz)	Clock ↑ to Data Float Delay	—	90	—	90	—	80
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	—	110	—	80	—	70
44	TdCr(Az)	Clock ↑ to Address Float Delay	—	110	—	90	—	80
45	TdCTr(A)	Address Stable after $\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑	160*		80*		35*	
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	90	—	60	—	60	—
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	—	0	—	0	—	0
48	TsINTf(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	80	—	80	—	70	—
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	—	0	—	0	—	0
50	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	920*		565*		365*	
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	—	110	—	85	—	70
52	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	100	—	85	—	70
53	TdCf(D)	Clock ↓ to Data Valid Delay	—	230	—	150	—	130

*For clock periods other than the minimums shown in the table, calculate parameters using the following expressions. Calculated values above assumed TrC = TfC = 20 ns.

Tabel 7/2.5-3: Schakeltijden (behorend bij de figuren 7/2.5-4 t/m 7/2.5-11).

2.5 Z80 (Z8400)

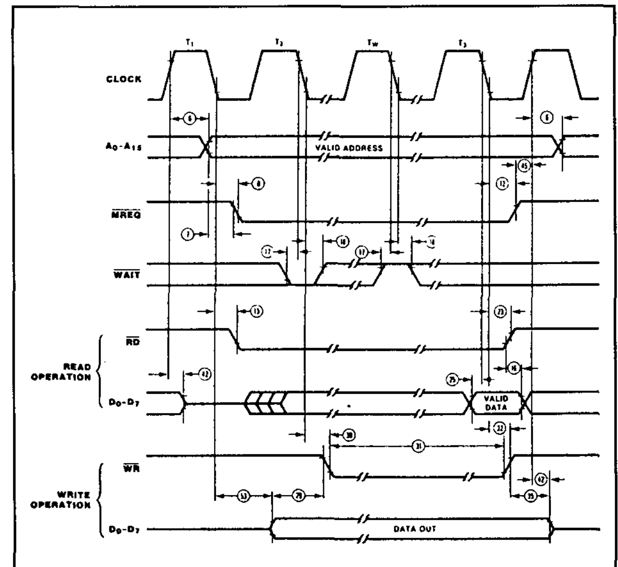
de adresbus (figuur 7/2.5-4). Ongeveer een halve klokcyclus later wordt MREQ aktief. De dalende flank van MREQ kan direct worden gebruikt als een Chip Enable voor dynamische geheugens. Wanneer RD aktief is, geeft dit aan dat de geheugen-data op de databus van de CPU kan worden gezet. De CPU bekijkt de WAIT-ingang bij de stijgende flank van de T3 klok. Tijdens de kloktoestanden T3 en T4 van een MI-cyclus kan verversen van de inhoud (refresh) van dynamische RAM's geschieden (Refresh Control = aktief), terwijl de CPU begint met het decoderen en uitvoeren van de instructie.



Figuur 7/2.5-4: Golfvormen en schakeltijden bij een Opcode Fetch Instructie (zie ook tabel 7/2.5-3).

Geheugen Lees- of Schrijf-cyclussen

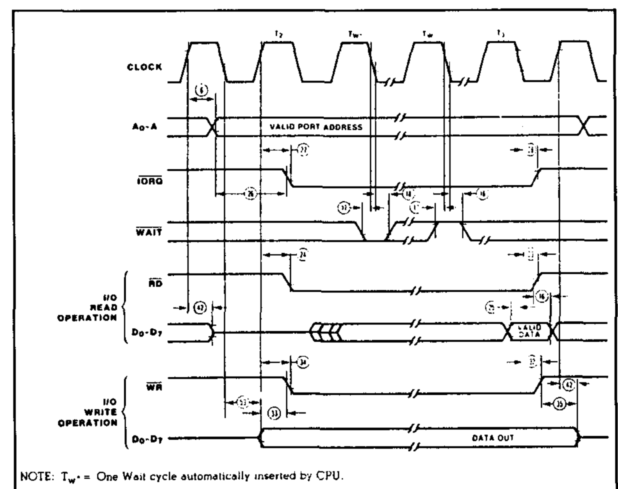
In figuur 7/2.5-5 is de timing van lees- of schrijfcyclussen met het geheugen te zien (geen opcode fetch (MI) cyclus). De MREQ en RD signalen werken precies zoals in de fetch cyclus. Bij een geheugen-schrijfcyclus wordt MREQ ook aktief als de adresbus stabiel is, zodat dit direct kan worden gebruikt als een Chip Enable voor dynamische RAM's. De WR-lijn is aktief wanneer de databus stabiel is, zodat die direct kan worden gebruikt als een R/W-puls voor de meeste halfgeleider-geheugens.



Figuur 7/2.5-5: Golfvormen en schakeltijden bij lezen uit of schrijven naar het geheugen (zie ook tabel 7/2.5-3).

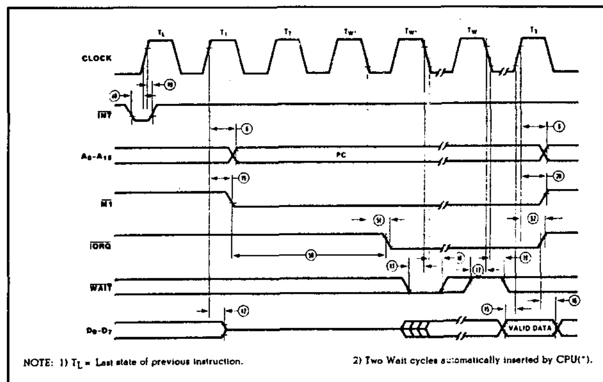
Ingangs- en Uitgangs- (I/O) Cyclussen

Figuur 7/2.5-6 laat de timing van een I/O-lees of een I/O-schrijf operatie zien. Tijdens de I/O-handelingen voegt de CPU automatisch een enkele WAIT-toestand (T_W) toe. Deze extra WAIT-toestand verschaft een I/O-poort voldoende tijd om het adres en de poortadreslijnen te decoderen.

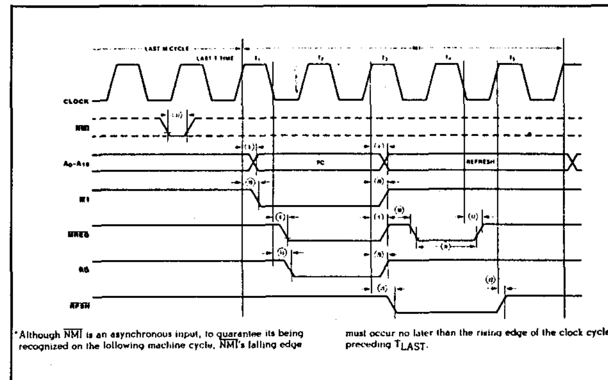


Figuur 7/2.5-6: Golfvormen en schakeltijden bij datatransport naar in- of uitgangen (zie ook tabel 7/2.5-3).

2.5 Z80 (Z8400)



Figuur 7/2.5-7: Golfvormen en schakeltijden bij een Interrupt Request/Acknowledge Cyclus (zie ook tabel 7/2.5-3).



Figuur 7/2.5-8: Golfvormen en schakeltijden bij de behandeling van een Niet-Maskeerbare Interrupt Request (zie tabel 7/2.5-3).

Interrupt Request/Acknowledge Cyclus

De CPU houdt op de stijgende flank van de laatste klokcyclus van elke instructie het interrupt-signaal in de gaten (figuur 7/2.5-7). Wordt een interruptie geaccepteerd dan wordt een speciale $\overline{MI}$-cyclus gegenereerd. Tijdens deze $\overline{MI}$-cyclus wordt $\overline{IORQ}$ aktief (in plaats van $\overline{MREQ}$) om aan te geven dat het interrumperende apparaat een 8-bits vector op de databus kan plaatsen. De CPU voegt automatisch twee WAIT-toestanden aan deze cyclus toe.

Niet-maskeerbare Interrupt Request Cyclus

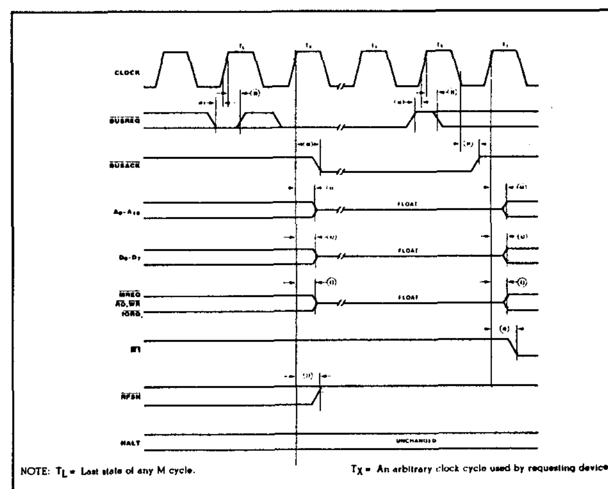
NMI wordt op dezelfde tijd als de maskeerbare interruptie $\overline{INT}$ afgevraagd, maar heeft een hogere prioriteit en kan niet onder software besturing worden geblokkeerd. De opvolgende timing komt overeen met die van een normale geheugen lees-operatie, behalve dat data die door het geheugen op de bus wordt gezet, wordt genegeerd. In plaats daarvan voert de CPU een herstart (RST) operatie uit en springt naar de $\overline{NMI}$ service-routine die zich op adres 0066H bevindt (figuur 7/2.5-8).

Bus Request/Acknowledge Cyclus

De CPU kijkt op de stijgende flank van de laatste klokperiode van elke machine-cyclus naar het $\overline{BUSREQ}$-signaal (figuur 7/2.5-9). Als $\overline{BUSREQ}$ aktief is, zet de CPU zijn adres, data, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ en $\overline{WR}$-lijnen op de stijgende flank van de volgende klokpuls in een hoog-impedante toestand. Op die tijd kan ieder extern apparaat de besturing van deze lijnen overnemen (meestal om data tussen geheugen en I/O-schakelingen over te brengen).

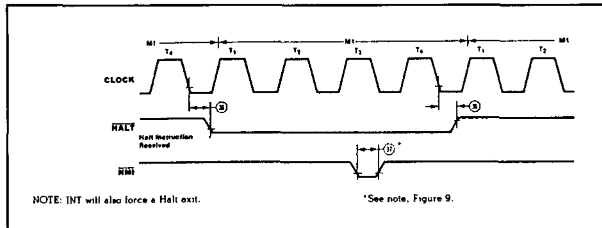
Halt Acknowledge Cyclus

Wanneer de CPU een $\overline{HALT}$-instructie ontvangt worden NOP toestanden uitgevoerd totdat een $\overline{INT}$ of een $\overline{NMI}$-signaal wordt ontvangen. In de Halt-toestand is de $\overline{HALT}$-



Figuur 7/2.5-9: Golfvormen en schakeltijden bij een Bus Request/Acknowledge Cyclus (zie ook tabel 7/2.5-3).

2.5 Z80 (Z8400)



Figuur 7/2.5-10: Golfvormen en schakeltijden bij een Halt Acknowledge Cyclus (zie tabel 7/2.5-3).

uitgang aktief en blijft zo totdat een interruptie is behandeld (figuur 7/2.5-10).

Reset Cyclus

$\overline{RESET}$ moet tenminste drie klokcyclussen aktief zijn om correct door de CPU te worden geaccepteerd. Zolang $\overline{RESET}$ aktief blijft, zweven de adres- en databussen en zijn de control-uitgangen niet-aktief. Op het niet-aktief worden van $\overline{RESET}$ worden twee interne T-cyclussen geconsumeerd voordat de CPU verder kan met de normale werking. $\overline{RESET}$ maakt het PC-register schoon zodat de eerste opcode fetch op lokatie 0000 plaatsvindt (figuur 7/2.5-11).

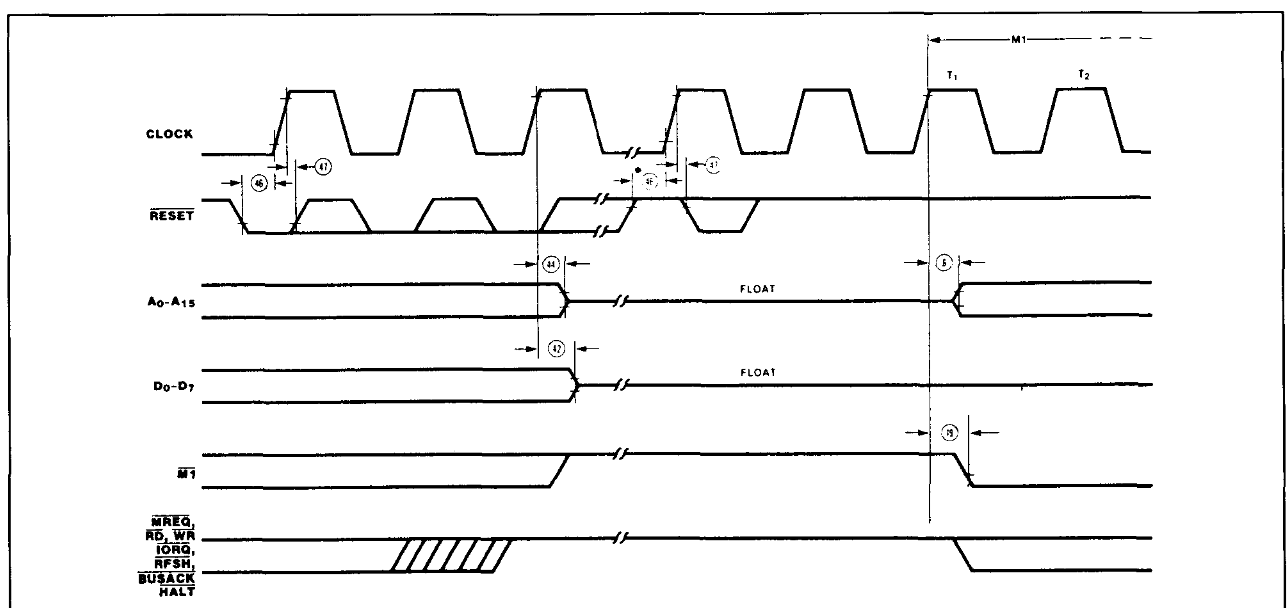
De instructie-set

De Z80 microprocessor heeft een zeer krachtige en veelzijdige instructie-set. Deze bevat unieke operaties zoals bijvoorbeeld block-move voor snelle, efficiënte data-overdrachten tussen geheugenplaatsen onderling of tussen geheugen en I/O. Hierdoor zijn ook bewerkingen van elk willekeurig bit op elke plaats in het geheugen mogelijk. In de tabellen 7/2.5-8 tot en met 7/2.5-19 en de bijbehorende figuren 7/2.5-12 tot en met 7/2.5-23 wordt een samenvatting van de instructie-set gegeven, met de mnemonische uitdrukking, de werking, de vlag-status en eventueel commentaar.

Overige elektrische kenmerken

In de tabellen 7/2.5-5, 6 en 7 zijn de overige elektrische grootheden van de Z80 microprocessor weergegeven.

Tenslotte zijn in de figuren 7/2.5-25 tot en met 7/2.5-28 nog enkele richtlijnen voor het programmeren van de parallelle In-



Figuur 7/2.5-11: Golfvormen en schakeltijden die optreden bij een Reset Cyclus (zie ook tabel 7/2.5-3).

2.5 Z80 (Z8400)

/Uitgangs-schakeling (PIO), Counter/Timer Circuit (CTC) en Seriële In-/Uitgangs schakeling (SIO) te zien.

		SOURCE															
		IMPLIED		REGISTER							REGISTER INDIRECT			INDEXED		EXT. ADDR.	IMME.
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	n
DESTINATION REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
REGISTER INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
INDEXED	(IX + d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY + d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
EXTERNAL ADDRESS	(nn)			32 n n													
IMPLIED	I			ED 47													
	R			ED 4F													

Figuur 7/2.5-12: Opcode-matrix voor de 8-bit Load-groep (zie ook tabel 7/2.5-9).

			SOURCE									
			REGISTER							IMM. EXT.	EXT. ADDR.	REG. INDIR.
			AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
DESTINATION	REGISTER	AF										F1
		BC								01 n n	ED 4B n n	C1
		DE								11 n n	ED 5B n n	D1
		HL								21 n n	2A n n	E1
		SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
		IX								DD 21 n n	DD 2A n n	DD E1
		IY								FD 21 n n	FD 2A n n	FD E1
	EXTERNAL ADDRESS	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS	REGISTER IND.	(SP)	F5	C5	D5	E5		DD E5	FD E5			

NOTE: The Push & Pop Instructions adjust the SP after every execution.

Figuur 7/2.5-13: Opcode-matrix voor de 16-bit Load-groep (zie ook tabel 7/2.5-10).

Exchange Group

		IMPLIED ADDRESSING				
		AF'	BC', DE' & HL'	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REGISTER INDIRECT	(SP)			E3	DD E3	FD E3

a

Block Transfer Group

			SOURCE	
			REG. INDIR.	
			(HL)	
DESTINATION	REG. INDIR	(DE)	ED A0	'LDI'—Load (DE) ← (HL) Inc HL & DE, Dec BC
			ED B0	'LDIR'—Load (DE) ← (HL) Inc HL & DE, Dec BC, Repeat until BC = 0
			ED A8	'LDD'—Load (DE) ← (HL) Dec HL & DE, Dec BC
			ED B8	'LDDR'—Load (DE) ← (HL) Dec HL & DE, Dec BC, Repeat until BC = 0

HL points to source
DE points to destination
BC is byte counter

b

Block Search Group

SEARCH LOCATION	
REG. INDIR.	
(HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR'—Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD'—Dec HL & BC
ED B9	'CPDR'—Dec HL & BC Repeat until BC = 0 or find match

HL points to location in memory to be compared with accumulator contents
BC is byte counter

c

Figuur 7/2.5-14: Opcode-matrix voor a) de Exchange-groep, b) de Transfer-groep en c) de Search-groep (zie ook tabel 7/2.5-11).

2.5 Z80 (Z8400)

	SOURCE										
	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

Figuur 7/2.5-15: Opcode-matrix voor de 8-bit Arithmetic- en de Logical-groep (zie ook tabel 7/2.5-12).

General-Purpose Arithmetic

a)

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

Miscellaneous CPU Control

b)

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM 0'	ED 46	8080A MODE
SET INT MODE 1 'IM 1'	ED 56	RESTART TO LOCATION 0038_H
SET INT MODE 2 'IM 2'	ED 5E	INDIRECT CALL USING REGISTER 1 AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

Figuur 7/2.5-16: Opcode-matrix voor a) de General-Purpose Arithmetic-groep en b) de Miscellaneous CPU Control-groep (zie tabel 7/2.5-13).

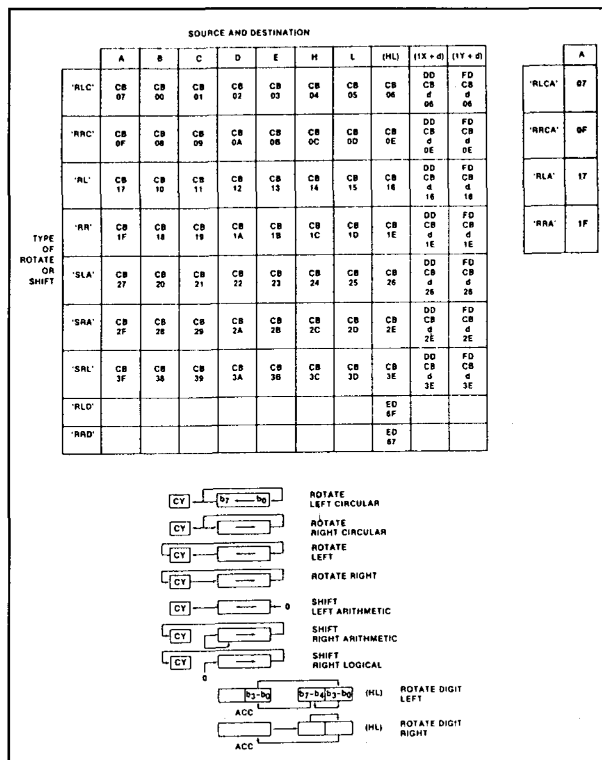
			SOURCE					
			BC	DE	HL	SP	IX	IY
DESTINATION	'ADD'	HL	09	19	29	39		
		IX	DD 09	DD 19		DD 39	DD 29	
		IY	FD 09	FD 19		FD 39		FD 29
	ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
	SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
	INCREMENT 'INC'		03	13	23	33	DD 23	FD 23
	DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

Figuur 7/2.5-17: Opcode-matrix voor de 16-bit Arithmetic-groep (zie ook tabel 7/2.5-14).

			CONDITION									
			UN COND	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B ≠ 0
JUMP 'JP'	IMMEDIATE EXTENSION	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JP'	RELATIVE	PC + e	18 e - 2	38 e - 2	30 e - 2	28 e - 2	20 e - 2					
JUMP 'JP'	REGISTER INDIRECT	(HL)	E9									
JUMP 'JR'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC·e										10 e - 2

Figuur 7/2.5-18: Opcode-matrix voor de Jump-groep (zie tabel 7/2.5-15).

2.5 Z80 (Z8400)

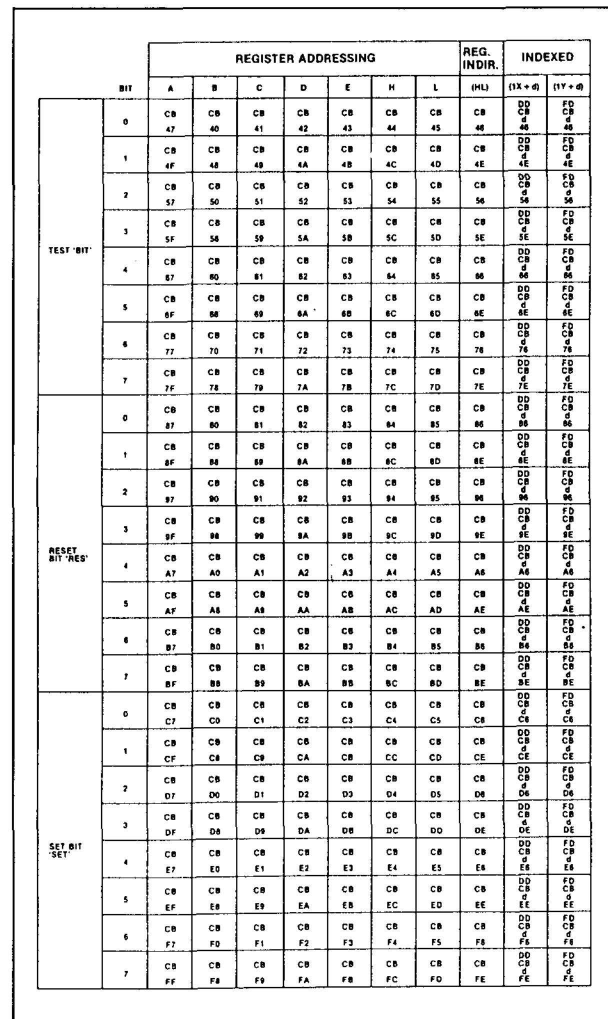


SOURCE AND DESTINATION

TYPE OF ROTATE OR SHIFT	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

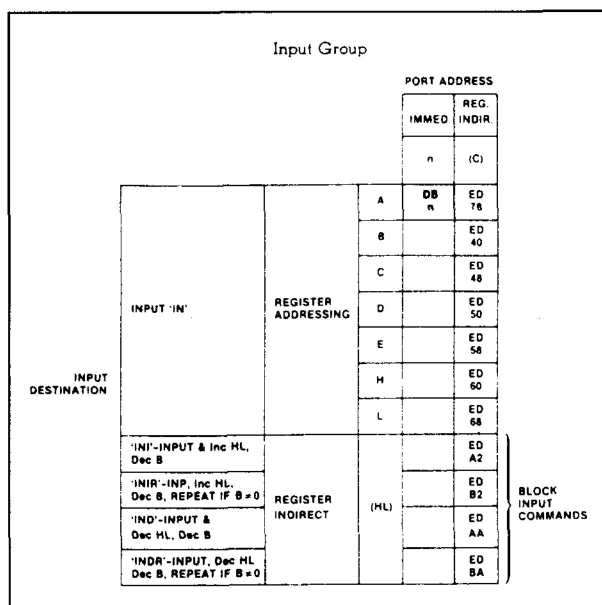
	A
'RLCA'	07
'RRCA'	0F
'RLA'	17
'RRA'	1F

Figuur 7/2.5-19: Opcode-matrix voor de Rotate- en Shift-groep (zie ook tabel 7/2.5-16).



	BIT	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
		REGISTER ADDRESSING							REG. INDIR.	INDEXED	
TEST 'BIT'	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

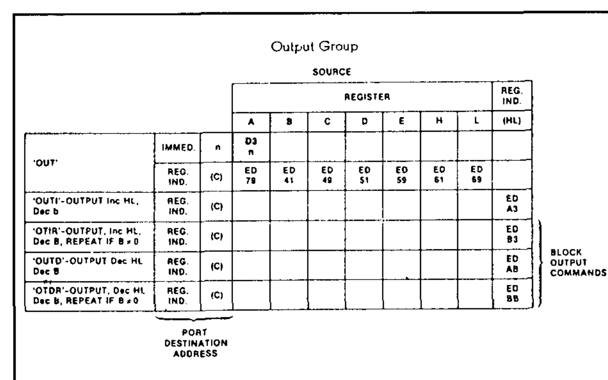
Figuur 7/2.5-20: Opcode-matrix voor de Bit Manipulatie-groep (zie tabel 7/2.5-17).



Input Group

INPUT DESTINATION				PORT ADDRESS IMMED. n	PORT ADDRESS REG. INDIR. (C)	
	INPUT 'IN'	REGISTER ADDRESSING	A	DB n	ED 78	
			B		ED 40	
			C		ED 48	
			D		ED 50	
			E		ED 58	
			H		ED 60	
			L		ED 68	
	'INI'-INPUT & Inc HL, Dec B	REGISTER INDIRECT	(HL)		ED A2	BLOCK INPUT COMMANDS
	'INIR'-INP, Inc HL, Dec B, REPEAT IF B ≠ 0				ED B2	
	'IND'-INPUT & Dec HL, Dec B				ED AA	
	'INDR'-INPUT, Dec HL Dec B, REPEAT IF B ≠ 0				ED BA	

Figuur 7/2.5-21: Opcode-matrix voor de Input-groep (zie ook tabel 7/2.5-18).



Output Group

SOURCE

	PORT DESTINATION ADDRESS		REGISTER A	REGISTER B	REGISTER C	REGISTER D	REGISTER E	REGISTER H	REGISTER L	REG. IND. (HL)	
'OUT'	IMMED.	n	D3 n								
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69		
'OUTI'-OUTPUT Inc HL, Dec b	REG. IND.	(C)								ED A3	
'OTIR'-OUTPUT, Inc HL, Dec B, REPEAT IF B ≠ 0	REG. IND.	(C)								ED B3	BLOCK OUTPUT COMMANDS
'OUTD'-OUTPUT Dec HL Dec B	REG. IND.	(C)								ED AB	
'OTDR'-OUTPUT, Dec HL Dec B, REPEAT IF B ≠ 0	REG. IND.	(C)								ED BB	

Figuur 7/2.5-22: Opcode-matrix voor de Output-groep (zie ook tabel 7/2.5-18).

2.5 Z80 (Z8400)

Call and Return Group

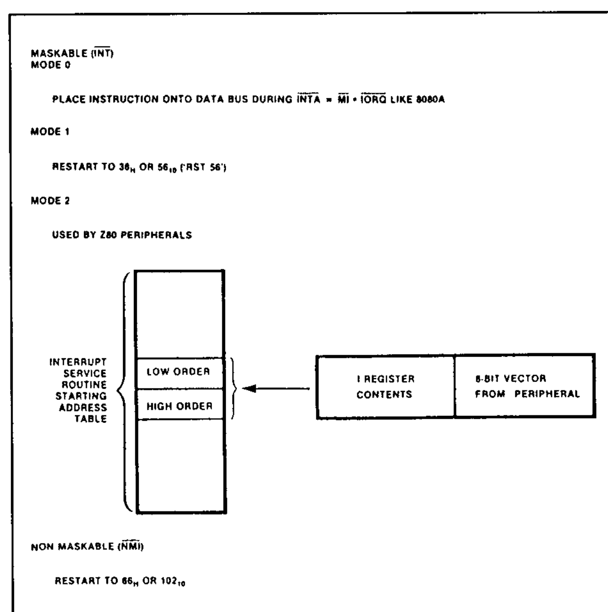
			CONDITION									
			UN COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG.	SIGN POS.	REG. B ≠ 0
'CALL'	IMMEDIATE EXTENSION	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
RETURN 'RET'	REGISTER INDIRECT	(SP) (SP + 1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REGISTER INDIRECT	(SP) (SP + 1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REGISTER INDIRECT	(SP) (SP + 1)	ED 45									

Note: Certain flags have more than one purpose.
Refer to the Z80 CPU Technical Manual for details.

Restart Group

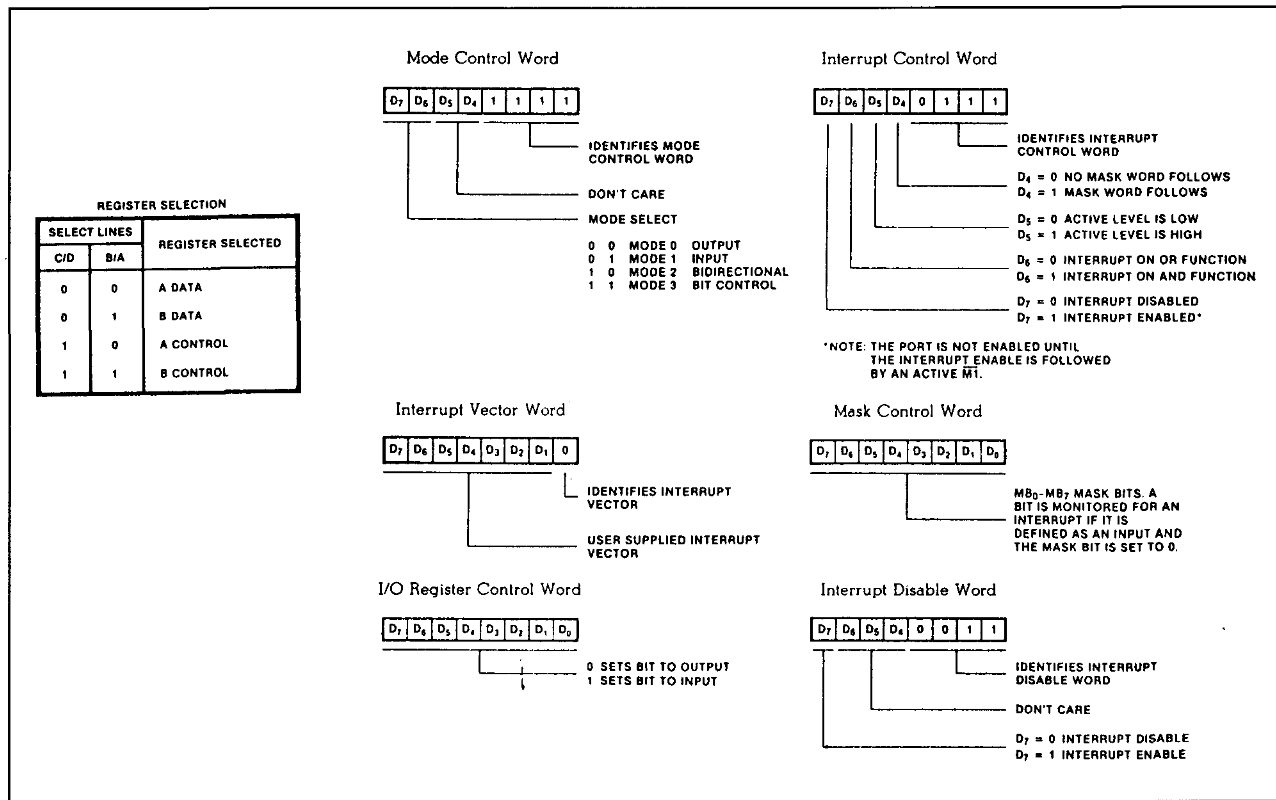
		OP CODE	
CALL ADDRESS	0000_H	C7	'RST 0'
	0008_H	CF	'RST 8'
	0010_H	D7	'RST 16'
	0018_H	DF	'RST 24'
	0020_H	E7	'RST 32'
	0028_H	EF	'RST 40'
	0030_H	F7	'RST 48'
	0038_H	FF	'RST 56'

Figuur 7/2.5-23: Opcode-matrix voor a) de Call en Return-groep en b) de Restart-groep (zie tabel 7/2.5-19).



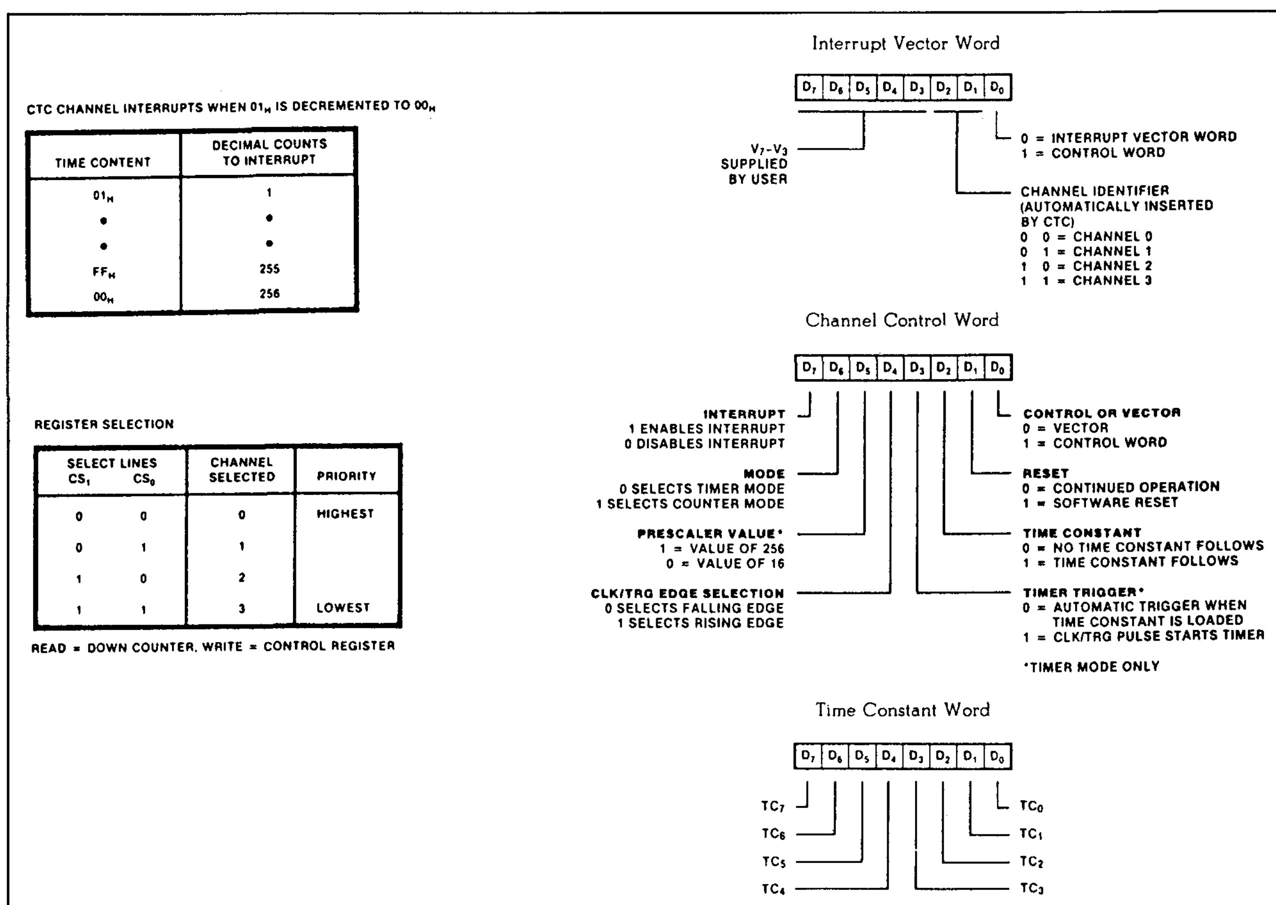
Figuur 7/2.5-24: Structuren bij maskeerbare en niet-maskeerbare interrupties.

2.5 Z80 (Z8400)



SELECT LINES C/D	SELECT LINES B/A	REGISTER SELECTED
0	0	A DATA
0	1	B DATA
1	0	A CONTROL
1	1	B CONTROL

Figuur 7/2.5-25: Samenvatting en Registers bij het programmeren van een PIO.

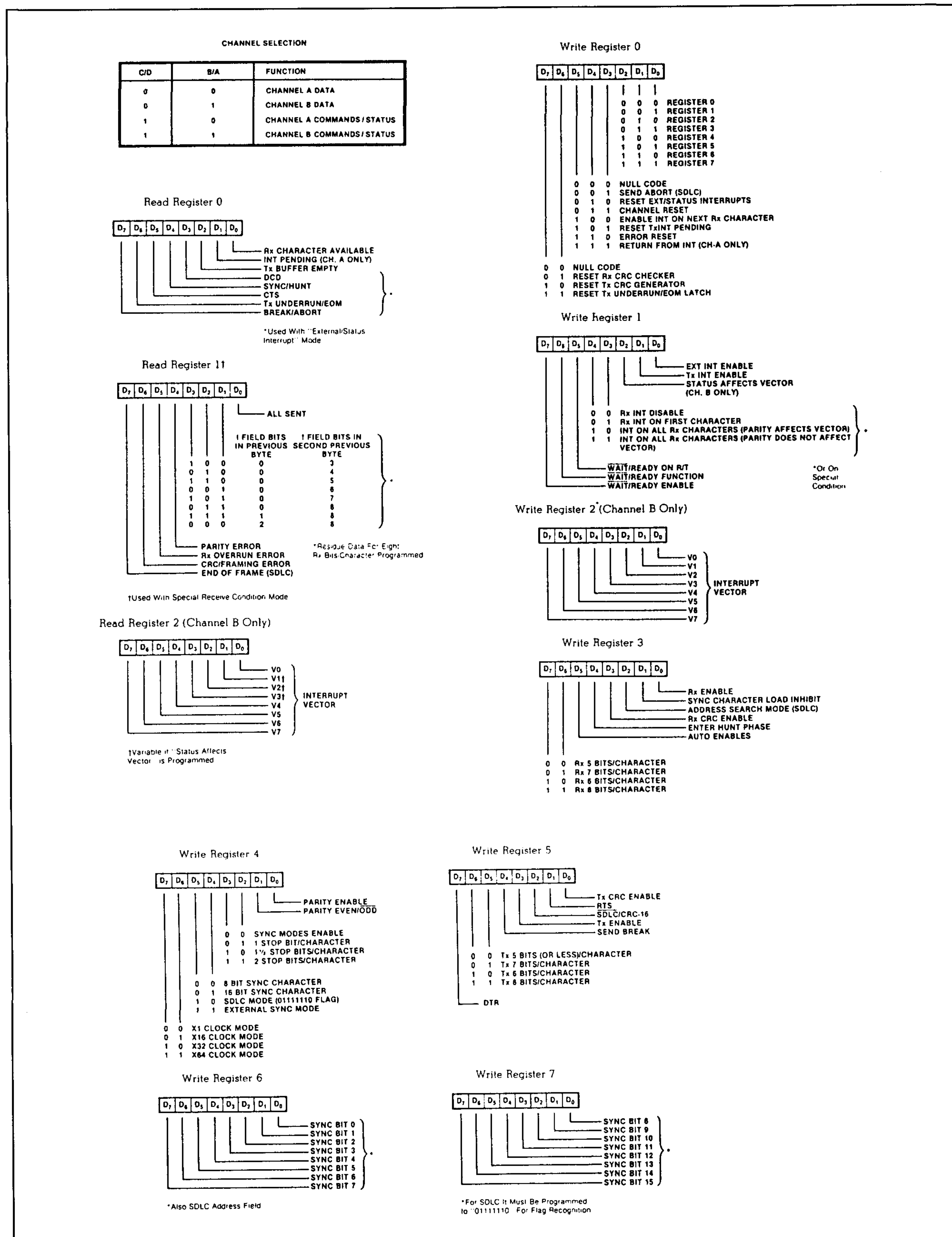


TIME CONTENT	DECIMAL COUNTS TO INTERRUPT
01_H	1
•	•
•	•
FF_H	255
00_H	256

SELECT LINES CS_1	SELECT LINES CS_0	CHANNEL SELECTED	PRIORITY
0	0	0	HIGHEST
0	1	1	
1	0	2	
1	1	3	LOWEST

Figuur 7/2.5-26: Samenvattingen en Registers bij het programmeren van een CTC.

2.5 Z80 (Z8400)



Figuur 7/2.5-27: Samenvatting en Registers bij het programmeren van een SIO.

2.5 Z80 (Z8400)

STATUS AFFECTS VECTOR (D_3) (FROM WRITE REGISTER 1)

IF THIS MODE IS SELECTED, THE VECTOR RETURNED FROM AN INTERRUPT ACKNOWLEDGE CYCLE WILL BE VARIABLE ACCORDING TO THE FOLLOWING:

	V_3	V_2	V_1	
CHANNEL B	0	0	0	CH B TRANSMIT BUFFER EMPTY
	0	0	1	CH B EXTERNAL/STATUS CHANGE
	0	1	0	CH B RECEIVE CHARACTER AVAILABLE
	0	1	1	CH B SPECIAL RECEIVE CONDITION
CHANNEL A	1	0	0	CH A TRANSMIT BUFFER EMPTY
	1	0	1	CH A EXTERNAL/STATUS CHANGE
	1	1	0	CH A RECEIVE CHARACTER AVAILABLE
	1	1	1	CH A SPECIAL RECEIVE CONDITION

IF THIS BIT IS 0, THE FIXED VECTOR PROGRAMMED IN THE VECTOR REGISTER IS RETURNED

Figuur 7/2.5-28: Vector na een Interrupt Acknowledge als de 'status beïnvloedt vector' mode is geselecteerd.

Absolute Maximum Ratings	Storage Temperature . . . . -65°C to +150°C
	Temperature under Bias . . . . Specified operating range
	Voltages on all inputs and outputs with respect to ground . -0.3 V to +7 V
	Power Dissipation . . . . 1.5 W

Tabel 7/2.5-5: Maximaal toegelaten waarden.

DC Characteristics	Symbol	Parameter	Min	Max	Unit	Test Condition
	V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
	V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
	V_{IL}	Input Low Voltage	-0.3	0.8	V	
	V_{IH}	Input High Voltage	2.0	V_{CC}	V	
	V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 1.8 mA
	V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 μA
	I_{CC}	Power Supply Current				
		Z80		150[1]	mA	
		Z80A		200[2]	mA	
		Z80B		200	mA	
	I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
	I_{LEAK}	3-State Output Leakage Current in Float	-10	10[3]	μA	V_{OUT} = 0.4 to V_{CC}

1. For military grade parts, I_{CC} is 200 mA.
2. Typical rate for Z80A is 90 mA.
3. A_{15}-A_0, D_7-D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$.

Tabel 7/2.5-6: Gelijkstroom eigenschappen van de Z8400.

Capacitance	Symbol	Parameter	Min	Max	Unit	Note
	C_{CLOCK}	Clock Capacitance		35	pF	Unmeasured pins returned to ground
	C_{IN}	Input Capacitance		5	pF	
	C_{OUT}	Output Capacitance		10	pF	

T_A = 25°C, f = 1 MHz.

Tabel 7/2.5-7: Capaciteiten bij 1 MHz.

2.5 Z80 (Z8400)

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	↕	X	V	0	↕	8-bit add or add with carry
SUB s; SBC A, s; CP s; NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s, XOR s	↕	↕	X	0	X	P	0	0	Logical operations
INC s	↕	↕	X	↕	X	V	0	•	8-bit increment
DEC s	↕	↕	X	↕	X	V	1	•	8-bit decrement
ADD DD, ss	•	•	X	X	X	•	0	↕	16-bit add
ADC HL, ss	↕	↕	X	X	X	V	0	↕	16-bit add with carry
SBC HL, ss	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry
RLA, RLCA, RRA, RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator
RL m, RLC m, RR m, RRC m, SLA m, SRA m, SRL m	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD, RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	↕	Complement carry
IN r (C)	↕	↕	X	0	X	P	0	•	Input register indirect
INI, IND, OUTI, OUTD	X	↕	X	X	X	X	1	•	Block input and output Z = 0 if B ≠ 0 otherwise Z = 0
INIR, INDR, OTIR, OTDR	X	1	X	X	X	X	1	•	Block input and output Z = 0 if B ≠ 0 otherwise Z = 0
LDI, LDD	X	X	X	0	X	↕	0	•	Block transfer instructions P/V = 1 if BC ≠ 0, otherwise P/V = 0
LDIR, LDDR	X	X	X	0	X	0	0	•	Block transfer instructions P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	X	↕	X	X	X	↕	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1
Z	Zero flag. Z = 1 if the result of the operation is 0
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator
N	Add/Subtract flag. N = 1 if the previous operation was a subtract
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation
0	The flag is reset by the operation
1	The flag is set by the operation
X	The flag is a "don't care"
V	P/V flag affected according to the overflow result of the operation
P	P/V flag affected according to the parity result of the operation
r	Any one of the CPU registers A, B, C, D, E, H, L
s	Any 8-bit location for all the addressing modes allowed for the particular instruction
ss	Any 16-bit location for all the addressing modes allowed for that instruction
ii	Any one of the two index registers IX or IY
R	Refresh counter
n	8-bit value in range < 0, 255 >
nn	16-bit value in range < 0, 65535 >

Tabel 7/2.5-8: Samenvatting van de Flag Operaties.

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments	
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r'	Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110		2	2	7	000	B
										– n –					001	C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010	D
LD r, (IX + d)	r ← (IX + d)	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	011	E
										01 r 101					100	H
										– d –					101	L
LD r, (IY + d)	r ← (IY + d)	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	111	A
										01 r 110						
										– d –						
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7		
LD (IX + d), r	(IX + d) ← r	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19		
										01 110 r						
										– d –						
LD (IY + d), r	(IY + d) ← r	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19		
										01 110 r						
										– d –						
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110	36	2	3	10		
										– n –						
LD (IX + d), n	(IX + d) ← n	•	•	X	•	X	•	•	•	11 011 101	DD	4	5	19		
										00 110 110	36					
										– d –						
										– n –						
LD (IY + d), n	(IY + d) ← n	•	•	X	•	X	•	•	•	11 111 101	FD	4	5	19		
										00 110 110	36					
										– d –						
										– n –						
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7		
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7		
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010	3A	3	4	13		
										– n –						
										– n –						
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7		
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7		
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010	32	3	4	13		
										– n –						
										– n –						
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9		
										01 010 111	57					
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9		
										01 011 111	5F					
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9		
										01 000 111	47					
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9		
										01 001 111	4F					

NOTES r, r' means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Tabel 7/2.5-9: Instructies in de 8-bit Load-groep (zie ook figuur 7/2.5-12).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD dd, nn	dd - nn	•	•	X	•	X	•	•	•	00 dd0 001 - n - - n -		3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX - nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 - n - - n -	DD 21	4	4	14	
LD IY, nn	IY - nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 - n - - n -	FD 21	4	4	14	
LD HL, (nn)	H - (nn + 1) L - (nn)	•	•	X	•	X	•	•	•	00 101 010 - n - - n -	2A	3	5	16	
LD dd, (nn)	dd_H - (nn + 1) dd_L - (nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 - n - - n -	ED	4	6	20	
LD IX, (nn)	IX_H - (nn + 1) IX_L - (nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 - n - - n -	DD 2A	4	6	20	
LD IY, (nn)	IY_H - (nn + 1) IY_L - (nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 - n - - n -	FD 2A	4	6	20	
LD (nn), HL	(nn + 1) - H (nn) - L	•	•	X	•	X	•	•	•	00 100 010 - n - - n -	22	3	5	16	
LD (nn), dd	(nn + 1) - dd_H (nn) - dd_L	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 - n - - n -	ED	4	6	20	
LD (nn), IX	(nn + 1) - IX_H (nn) - IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 - n - - n -	DD 22	4	6	20	
LD (nn), IY	(nn + 1) - IY_H (nn) - IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 - n - - n -	FD 22	4	6	20	
LD SP, HL	SP - HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP - IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP - IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP - 2) - qq_L (SP - 1) - qq_H SP - SP - 2	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP - 2) - IX_L (SP - 1) - IX_H SP - SP - 2	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP - 2) - IY_L (SP - 1) - IY_H SP - SP - 2	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_H - (SP + 1) qq_L - (SP) SP - SP + 2	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H - (SP + 1) IX_L - (SP) SP - SP + 2	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H - (SP + 1) IY_L - (SP) SP - SP + 2	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

NOTES dd is any of the register pairs BC DE HL SP
qq is any of the register pairs AF BC DE HL
$(PAIR)_H$ $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively
e g BC_L = C AF_H = A

Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-10: Instructies in de 16-bit Load groep (zie ook figuur 7/2.5-13).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
EX DE, HL	DE – HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF – AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	BC – BC' DE – DE' HL – HL'	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H – (SP + 1) L – (SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_H – (SP + 1) IX_L – (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_H – (SP + 1) IY_L – (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE) – (HL) DE – DE + 1 HL – HL + 1 BC – BC – 1	•	•	X	0	X	① ↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) – (HL) DE – DE + 1 HL – HL + 1 BC – BC – 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) – (HL) DE – DE – 1 HL – HL – 1 BC – BC – 1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE) – (HL) DE – DE – 1 HL – HL – 1 BC – BC – 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A – (HL) HL – HL + 1 BC – BC – 1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A – (HL) HL – HL + 1 BC – BC – 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A – (HL) HL – HL – 1 BC – BC – 1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A – (HL) HL – HL – 1 BC – BC – 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES ① P/V flag is 0 if the result of BC – 1 = 0, otherwise P/V = 1

② Z flag is 1 if A = (HL), otherwise Z = 0

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-11: Instructies in de Exchange-, Block Transfer- en Search groepen (zie ook figuur 7/2.5-14).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 [000] r		1	1	4	r Reg
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 [000] 110		2	2	7	000 B
										– n –					001 C
															010 D
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10 [000] 110		1	2	7	011 E
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X	↕	X	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 [000] 110					101 L
										– d –					111 A
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X	↕	X	V	0	↕	11 111 101	FD	3	5	19	
										10 [000] 110					
										– d –					
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕	[001]					s is any of r, n,
SUB s	A ← A – s	↕	↕	X	↕	X	V	1	↕	[010]					(HL), (IX + d),
SBC A, s	A ← A – s – CY	↕	↕	X	↕	X	V	1	↕	[011]					(IY + d) as shown
AND s	A ← A s	↕	↕	X	1	X	P	0	0	[100]					for ADD instruction
OR s	A ← A s	↕	↕	X	0	X	P	0	0	[110]					The indicated bits
XOR s	A ← A ● s	↕	↕	X	0	X	P	0	0	[101]					replace the [000] in
CP s	A – s	↕	↕	X	↕	X	V	1	↕	[111]					the ADD set above.
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r [100]		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	00 110 [100]		1	3	11	
INC (IX + d)	(IX + d) ←	↕	↕	X	↕	X	V	0	•	11 011 101	DD	3	6	23	
	(IX + d) + 1									00 110 [100]					
										– d –					
INC (IY + d)	(IY + d) ←	↕	↕	X	↕	X	V	0	•	11 111 101	FD	3	6	23	
	(IY + d) + 1									00 110 [100]					
										– d –					
DEC m	m ← m – 1	↕	↕	X	↕	X	V	1	•	[101]					m is any of r, (HL), (IX + d), (IY + d) as shown for INC. DEC same format and states as INC. Replace [100] with [101] in opcode

NOTES The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow. P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Tabel 7/2.5-12: Instructies in de 8-bit Arithmetic en Logical groep (zie ook figuur 7/2.5-15).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (one's complement).
NEG	A ← 0 – A	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement)
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

NOTES: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.
★ indicates interrupts are not sampled at the end of EI or DI

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-13: Instructies in de General-Purpose Arithmetic en CPU Control groepen (zie ook figuur 7/2.5-16).

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL ← HL – ss – CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 01 pp1 001	DD	2	4	15	pp Reg 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Reg 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss ← ss – 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX – 1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY – 1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

NOTES: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-14: Instructies in de 16-bit Arithmetic groep (zie ook figuur 7/2.5-17).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →		3	3	10	cc Condition 000 NZ non zero 001 Z zero 010 NC non carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 ← e − 2 →	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e − 2 →	38	2	2	7	If condition not met
	If C = 1, PC ← PC + e											2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e − 2 →	30	2	2	7	If condition not met
	If C = 0, PC ← PC + e											2	3	12	If condition is met
JP Z, e	If Z = 0, continue	•	•	X	•	X	•	•	•	00 101 000 ← e − 2 →	28	2	2	7	If condition not met
	If Z = 1, PC ← PC + e											2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 ← e − 2 →	20	2	2	7	If condition not met
	If Z = 0, PC ← PC + e											2	3	12	If condition is met
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B − 1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e − 2 →	10	2	2	8	If B = 0
	If B ≠ 0, PC ← PC + e											2	3	13	If B ≠ 0

NOTES e represents the extension in the relative addressing mode
e is a signed two's complement number in the range < −126, 129 >
e − 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-15: Instructies in de Jump groep (zie ook figuur 7/2.5-18).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RLCA	CY ← 7 ← 0 ← (A)	•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA	CY ← 7 ← 0 ← (A)	•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA	7 → 0 → CY (A)	•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA	7 → 0 → CY (A)	•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 [000] r	CB	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 [000] 110	CB	2	4	15	r Reg 000 B 001 C
RLC (IX + d)	CY ← 7 ← 0 ← r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 – d – 00 [000] 110	DD CB	4	6	23	010 D 011 E 100 H 101 L 111 A
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 – d – 00 [000] 110	FD CB	4	6	23	
RL m	CY ← 7 ← 0 ← m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	[010]					Instruction format and states are as shown for RLC's. To form new opcode replace [000] or RLC's with shown code
RRC m	7 → 0 → CY m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	[001]					
RR m	7 → 0 → CY m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	[011]					
SLA m	CY ← 7 ← 0 ← 0 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	[100]					
SRA m	7 → 0 → CY m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	[101]					
SRL m	0 → 7 → 0 → CY m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	[111]					
RLD	7-4 3-0 (A) ↔ 7-4 3-0 (HL)	↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL).
RRD	7-4 3-0 (A) ↔ 7-4 3-0 (HL)	↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	The content of the upper half of the accumulator is unaffected.

NOTES e represents the extension in the relative addressing mode
e is a signed two's complement number in the range < -126, 129 >.
e-2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-16: Instructies in de Rotate en Shift groep (zie ook figuur 7/2.5-19).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
BIT b, r	$Z \leftarrow \overline{r}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, $(IX+d)_b$	$Z \leftarrow (\overline{IX+d})_b$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b, $(IY+d)_b$	$Z \leftarrow (\overline{IY+d})_b$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 [11] b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 [11] b 110	CB	2	4	15	
SET b, (IX + d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 – d – [11] b 110	DD CB	4	6	23	
SET b, (IY + d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 – d – [11] b 110	FD CB	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX + d) (IY + d)	•	•	X	•	X	•	•	•	[10]					To form new opcode replace [11] of SET b, s with [10]. Flags and time states for SET instruction.

NOTES: The notation m_b indicates bit b (0 to 7) or location m.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Tabel 7/2.5-17: Instructies in de Bit Manipulation groep (zie ook figuur 7/2.5-20).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B − 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 010	ED A2	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
IND	(HL) ← (C) B ← B − 1 HL ← HL − 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 010	ED AA	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011 ← n →	D3	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B − 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTD	(C) ← (HL) B ← B − 1 HL ← HL − 1	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

NOTE ① If the result of B − 1 is zero the Z flag is set, otherwise it is reset

Flag Notation • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-18: Instructies in de Input en Output groep (zie ook de figuren 7/2.5-21 en 7/2.5-22).

2.5 Z80 (Z8400)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
CALL nn	(SP − 1) ← PC_H (SP − 2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 001 101 − n − − n −	CD	3	5	17	
CALL cc, nn	If condition cc is false continue otherwise same as CALL nn	•	•	X	•	X	•	•	•	11 cc 100 − n − − n −		3 3	3 5	10 17	If cc is false If cc is true
RET	PC_L ← (SP) PC_H ← (SP + 1)	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	•	X	•	•	•	11 cc 000		1 1	1 3	5 11	If cc is false If cc is true cc Condition 000 NZ non-zero 001 Z zero 010 NC non carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	(SP − 1) ← PC_H (SP − 2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	•	X	•	•	•	11 t 111		1	3	11	t p 000 00H 001 08H 010 10H 011 18H 100 20H 101 28H 110 30H 111 38H

NOTE RETN loads $IFF_2 \rightarrow IFF_1$

Flag Notation • = flag not affected 0 = flag reset 1 = flag set X = flag is unknown
↕ = flag is affected according to the result of the operation

Tabel 7/2.5-19: Instructies in de Call en Return groepen en Restart (zie ook figuur 7/2.5-23).

2.5 Z80 (Z8400)

7/3

Zestien bits processoren

Inhoud

3.1 8086

7/3.1

8086

Algemene gegevens

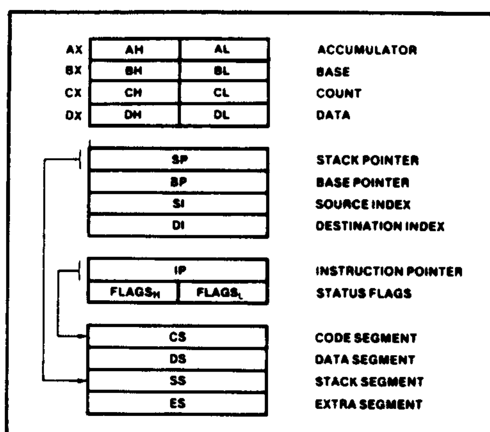
Versies

Intel: 8086, 8086-1, 8086-2, 80C86, 80C86-2
Siemens: SAB 8086, SAB 8086-1, SAB 8086-2
AMD: 8086, 8086-1, 8086-2
Fujitsu: MBL 8086, MBL 8086-1, MBL 8086-2
Mitsubishi: M5L 8086
NEC: uPD 8086, uPD 8086-2
Harris: 80C86

Inleiding: verschillen tussen 8086 en 8088

De 8086 is een 16 bits industriestandaard microprocessor die evenals de 8088 in veel IBM-PC compatible 'klonen' wordt toegepast. Aangezien de 8086 volledig 16 bits is, is de 'performance' van de PC's waarin de 8086 wordt toegepast gewoonlijk beter (sneller) dan die met de 8088 (pseudo-16 bit).

De meeste interne functies van de 8088 zijn identiek aan die van de 8086. De externe bus wordt door de 8088 op dezelfde manier behandeld als de 8086, met het verschil dat de 8088 slechts 8 bits tegelijk doet. Ook de interne register structuur is gelijk (tabel 7/3.1-1) en alle instructies leiden tot vergelijkbare resultaten.



Tabel 7/3.1-1: Overzicht van de interne registers.

Intern zijn er drie verschillen tussen de 8088 en de 8086, die alle betrekking hebben op de 8-bit bus interface van de 8088.

- De lengte van de wachtrij (queue) is 4 bytes in de 8088, terwijl de wachtrij van 8086 6 bytes of drie woorden kan bevatten.
- Om de wachtrij verder te optimaliseren werd in de 8088 de prefetch algorithme gewijzigd. De BIU van de 8088 haalt een nieuwe instructie op voor de wachtrij telkens wanneer deze ruimte heeft voor 1 byte. De 8086 wacht tot er ruimte is voor 2 bytes.
- De interne executietijd van de 8088 is aangetast door de 8 bit interface. Voor alle 16 bit ophaal- en schrijfhandelingen van/naar het geheugen zijn vier extra CLK-cyclussen nodig.

Door hun identieke executie-eenheden zijn de 8088 en 8086 volledig software compatibel, met uitzondering van die software die systeemafhankelijk is.

De belangrijkste verschillen tussen de twee processoren hebben te maken met de hardware. De functies van de aansluitpennen zijn bijna gelijk, met de volgende uitzonderingen.

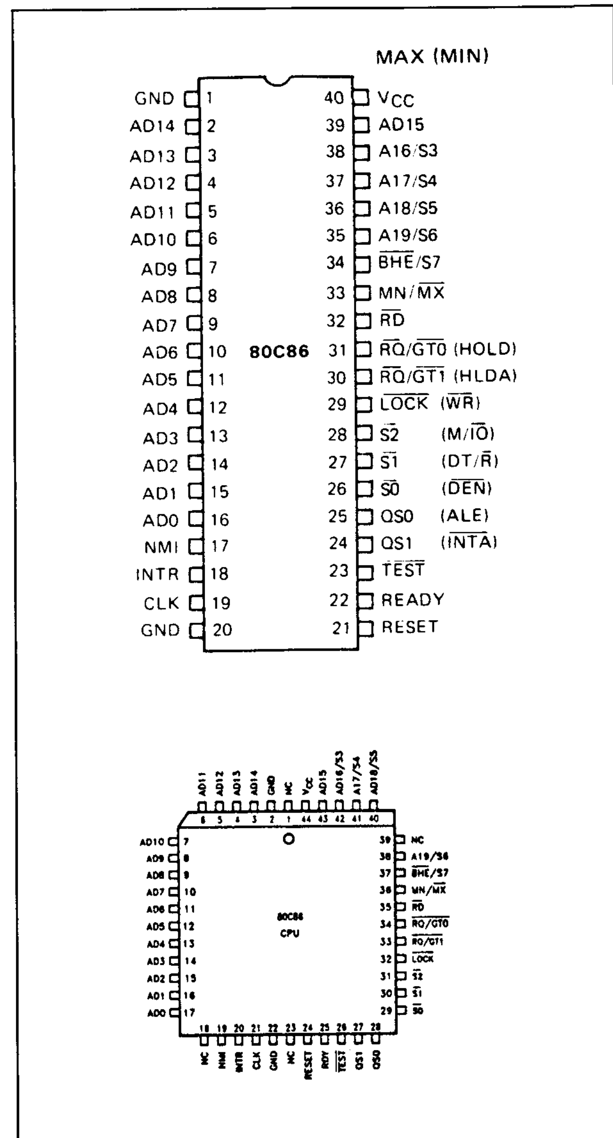
- A15 t/m A8 - Dit zijn bij de 8088 alleen intern gelatchte adresuitgangen.

3.1 8086

- $\overline{BHE}$ heeft bij de 8088 geen betekenis en is weggelaten.
- Bij de 8088 levert $\overline{SSO}$ (op pen 34) de $\overline{SO}$ statusinformatie in de minimum mode.
- IO/$\overline{M}$ bij de 8088 is geïnverteerd om compatibel te zijn met de MCS-85 bus structuur (M/$\overline{IO}$ bij 8086).
- ALE van de 8088 is in de minimum mode met één CLK-cyclus vertraagd om bij HALT de status de gelegenheid te geven om gelatcht te worden met ALE.

Omdat de 8086 de basis vormt van een gehele familie microprocessoren (80186, 80286, 80386) wordt ook deze microprocessor hier uitvoerig behandeld.
De 8086 kan werken in twee modes: MINImum voor kleine systemen en MAXimum voor grotere toepassingen zoals multi-processing.

De belangrijkste kenmerken
- Directe adressering tot 1 Mbyte geheugen;
- 14 woorden van 16-bit registerset met symmetrische werking;
- 24 operand adresseermodes;
- bit, byte, woord en blok handelingen mogelijk;
- 8-bit en 16-bit rekenen met en zonder teken, binair of decimaal, inclusief vermenigvuldigen en delen;
- verkrijgbaar in N-kanaals MOS en CMOS uitvoering;
- verschillende kloksnelheden:
  5 MHz voor 8086 (80C86);
  8 MHz voor 8086-2 (80C86-2);
  10 MHz voor 8086-1;
- MULTIBUS systeem compatibel;
- verkrijgbaar in EXPRESS (standaard) of uitgebreid temperatuurbereik;
- 40-pens DIL-behuizing, CMOS ook in 44-pens PLCC (figuur 7/3.1-1).



Figuur 7/3.1-1: Aansluitgegevens van de 8086 processor in 40-pens DIL en 44-pens PLCC-uitvoering.

Definities en functies van de aansluitpennen

De hierna volgende functies van de aansluitpennen gelden voor de 8086 in zowel de minimum- als de maximum mode. De hierin genoemde 'lokale bus' is de direct gemultiplexte interface-verbinding met de 8086 (waarbij geen rekening wordt gehouden met externe bus-buffers).

AD15–AD0: Adres Data Bus
(Input/Output, 3-state)
Deze lijnen vormen de tijd-gemultiplexte geheugen/IO adresbus (T1) en databus (T2,

3.1 8086

T3, Tw en T4). A0 komt overeen met $\overline{BHE}$ voor het laagste byte van de databus (D7 tot en met D0). A0 is LAAG tijdens T1 wanneer bij geheugen- of I/O-bewerkingen een byte moet worden overgebracht in het laagste gedeelte van de bus. Acht-bits schakelingen die alleen op de laagste helft worden aangesloten, gebruiken A0 gewoonlijk voor chip-select functies (zie $\overline{BHE}$). Deze lijnen zijn aktief-HOOG en komen gedurende interrupt acknowledge en lokale bus 'hold acknowledge' in de hoog-impedante toestand.

A19/S6, A18/S5, A17/S4, A16/S3: Adres/ Status

(Output, 3-state)

Gedurende T1 zijn dit de vier belangrijkste adreslijnen voor geheugen-operaties. Tijdens I/O-operaties zijn deze lijnen LAAG. Bij geheugen- en I/O-handelingen is statusinformatie hierop beschikbaar tijdens T2, T3, Tw en T4. S6 is altijd LAAG. De status van het interrupt enable-vlagbit (S5) wordt aan het begin van elke klokcyclus bijgewerkt. A17/S4 en A16/S3 worden volgens tabel 7/3.1-2 gecodeerd. Deze informatie geeft aan welk

S_4	S_3	CHARACTERISTICS
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

Tabel 7/3.1-2: Codering van het gebruikte segmentregister met S3 en S4.

relokatie-register op dat moment wordt gebruikt om toegang tot de data te krijgen. Tijdens lokale bus 'hold acknowledge' zijn deze lijnen hoog-impedant.

$\overline{BHE}$: Bus High Enable/Status

(Output, 3-state)

Gedurende T1 moet het bus high enable-signaal ($\overline{BHE}$) worden gebruikt om data vrij te geven voor de hoogste helft van de databus (D15 tot en met D8). Acht-bits schakelingen die op de hoogste helft van de databus worden aangesloten, gebruiken $\overline{BHE}$ gewoonlijk voor chip select functies. $\overline{BHE}$ is LAAG tijdens T1 voor lees-, schrijf- en interrupt-acknowledge-cyclussen wanneer een byte naar het hoogste gedeelte van de bus moet worden overgebracht. De S7 statusinformatie is beschikbaar gedurende T2, T3 en T4. Het signaal is aktief-LAAG en komt tijdens 'hold' in de hoog-impedante toestand. Het is LAAG tijdens T1 voor de eerste interrupt acknowledge cyclus (zie tabel 7/3.1-3).

$\overline{BHE}$	A_0	Characteristics
0	0	Whole word
0	1	Upper byte from/ to odd address
1	0	Lower byte from/ to even address
1	1	None

Tabel 7/3.1-3: Selectie van de juiste byte(s) van het geheugen/IO-woord met BHE en A0.

$\overline{RD}$: Read

(Output, 3-state)

De read-strobe geeft aan dat de processor bezig is met een geheugen- of I/O-leescyclus, afhankelijk van de toestand van de S2-pen. Het signaal wordt gebruikt om schakelingen die met de lokale 8086-bus zijn verbonden in te lezen. $\overline{RD}$ is aktief-LAAG tijdens T2, T3 en Tw van elke leescyclus en blijft gegarandeerd HOOG in T2 totdat de lokale 8086-bus heeft gezweefd. De uitgang is hoog-impedant tijdens 'hold acknowledge'.

READY: Ready

(Input)

READY is de bevestiging van het geadresseerde geheugen of I/O-schakeling dat het de data-overdracht zal afmaken. Het READY signaal van geheugen of I/O wordt door de 8284A clock generator gesynchroniseerd om zo READY te vormen. Het signaal is aktief-HOOG. De READY ingang van de 8086 wordt niet gesynchroniseerd. Een juiste werking wordt alleen gegarandeerd als wordt voldaan aan de setup en hold tijden.

INTR: Interrupt Request

(Input)
Interrupt request is een niveau-getriggerde ingang die gedurende de laatste klokcyclus van elke instructie wordt afgetast om te bepalen of de processor een interrupt acknowledge-handeling moet uitvoeren. Via een vector-opzoektabel die in het geheugen staat wordt naar een subroutine gesprongen. De ingang kan intern worden gemaskeerd door het interrupt enable-bit met software te resetten. INTR wordt intern gesynchroniseerd en is aktief-HOOG.

$\overline{\text{TEST}}$: Test

(Input)
Deze ingang wordt bekeken door de 'wait' opdracht. Als de $\overline{\text{TEST}}$-ingang LAAG is, gaat de uitvoering door, anders wacht de processor in een 'stationaire' toestand. Deze ingang wordt intern gesynchroniseerd op de voorflank van CLK tijdens elke klokcyclus.

NMI: Non-Maskable Interrupt

(Input)
NMI is een flankgetriggerde ingang die een type 2 interruptie veroorzaakt. Via een vector-opzoektabel in het geheugen wordt naar een subroutine gesprongen. NMI kan niet intern worden gemaskeerd met software. Een LAAG-naar-HOOG overgang leidt de interruptie in op het einde van de lopende instructie. Deze ingang wordt intern gesynchroniseerd.

RESET: Reset

(Input)
RESET laat de processor onmiddellijk zijn lopende aktiviteit beëindigen. Het signaal moet minstens vier klokcyclussen aktief-HOOG zijn en start de uitvoering opnieuw wanneer het weer LAAG wordt. RESET wordt intern gesynchroniseerd.

CLK: Clock

(Input)
De klok verzorgt de basis-timing van de processor en de buscontroller. De klok is asymmetrisch met een aan/uit-verhouding (duty-cycle) van 33 % om een optimale timing mogelijk te maken.

Vcc: Voedingsspanning (+5 V)

De voedingsspanning moet +5 V, ±10 % bedragen.

GND: Aarde (0 V)

MN/$\overline{\text{MX}}$: Minimum/Maximum

(Input)
Bepaalt de mode waarin de processor moet werken. Beide modes worden hierna besproken.

Maximum mode pennen

De volgende functiebeschrijvingen gelden voor een 8086/8288 systeem in de maximum mode (MN/$\overline{\text{MX}}$ = GND).

$\overline{\text{S2}}$, $\overline{\text{S1}}$, $\overline{\text{S0}}$: Status

(Output, 3-state)
De status is aktief gedurende T4, T1 en T2 en wordt weer passief (1, 1, 1) tijdens T3 of Tw als READY HOOG is. De 8288 buscontroller gebruikt de status om alle benodigde geheugen- en I/O-besturingssignalen op te wekken. Elke verandering van $\overline{\text{S2}}$, $\overline{\text{S1}}$ of $\overline{\text{S0}}$ tijdens T4 wordt gebruikt om het begin van een nieuwe buscyclus aan te geven en de terugkeer naar de passieve toestand tijdens T3 of Tw betekent het einde van een buscyclus. Tijdens 'hold acknowledge' zijn deze uitgangen hoog-impedant. De codering van deze statuslijnen is te zien in tabel 7/3.1-4.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	CHARACTERISTICS
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Tabel 7/3.1-4: Codering van de statuslijnen S2, S1 en S0.

3.1 8086

$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$: Request/Grant

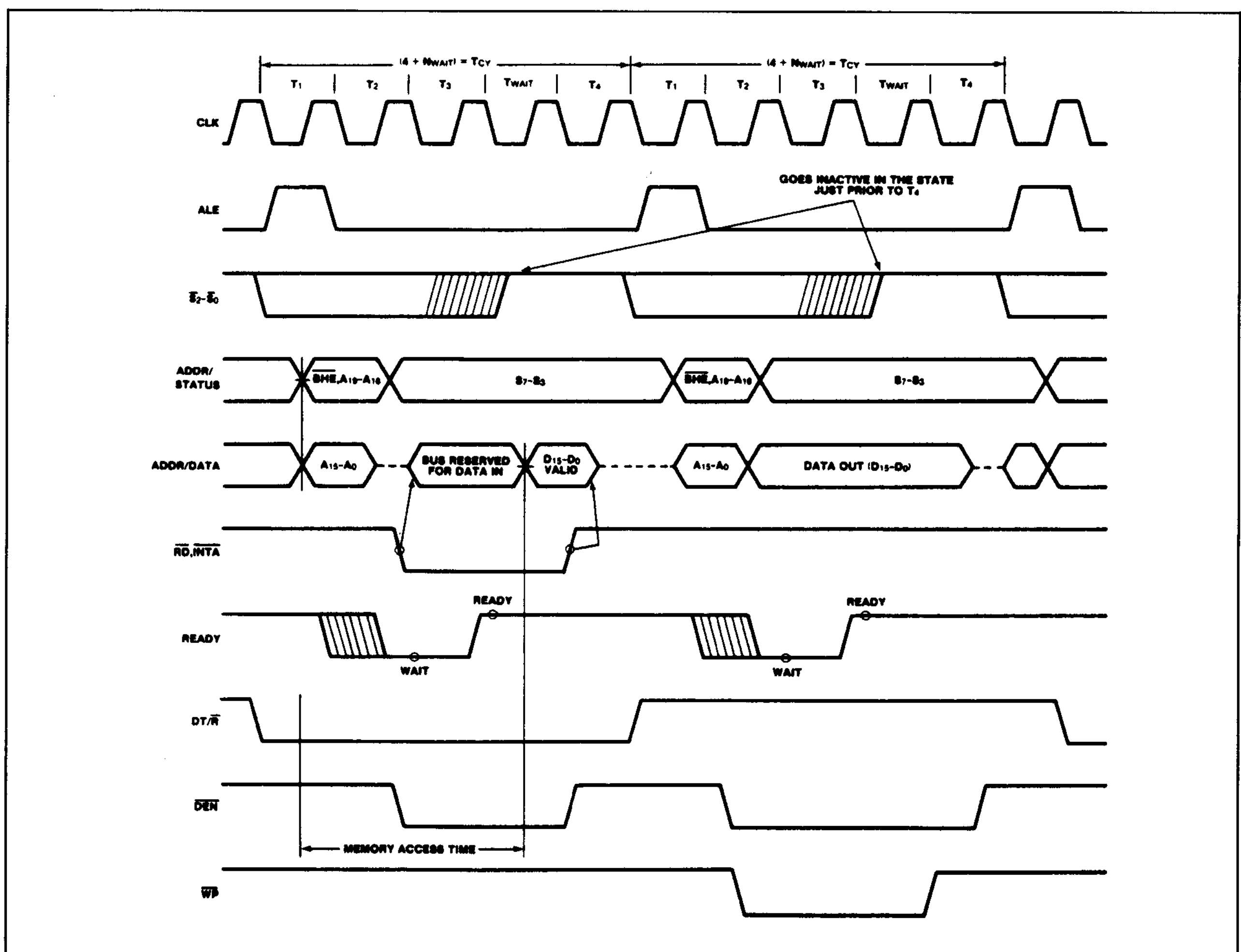
(Input/Output)

De request-grant pennen worden door andere lokale busmasters gebruikt om de processor te dwingen de lokale bus aan het einde van de lopende buscyclus vrij te geven. Elke pen is bidirectioneel, waarbij $\overline{RQ}/\overline{GT0}$ een hogere prioriteit heeft dan $\overline{RQ}/\overline{GT1}$. $\overline{RQ}/\overline{GT}$ heeft een interne optrekweerstand en mag dus los blijven.

De request/grant volgorde is als volgt (zie ook figuur 7/3.1-2).

1 - Een 1 CLK brede puls van een andere busmaster laat aan de 8086 een lokale bus-request ('hold') zien (puls 1).
2 - Gedurende een T4 of T1 klokcyclus laat de 8086 met een 1 CLK brede puls (puls 2) aan de vragende master weten dat hij heeft toegestaan dat de bus zwevend wordt en dat hij bij de volgende CLK in de 'hold acknowledge' toestand gaat. De bus-interface van de processor wordt tijdens 'hold acknowledge' logisch losgekoppeld van de lokale bus.
3 - Een 1 CLK brede puls van de vragende master laat aan de 8086 weten (puls 3) dat het 'hold' verzoek bijna ten einde is en dat de 8086 de lokale bus bij de volgende CLK weer kan opeisen.

Elke master-master wisseling van de lokale



Figuur 7/3.1-2: Principiële systeemtiming.

bus levert drie pulsen op. Na elke buswisseling moet een 'dode' CLK-cyclus komen. De pulsen zijn aktief-LAAG.

Wanneer het verzoek komt terwijl de processor een geheugencyclus uitvoert, wordt de lokale bus tijdens T4 van de cyclus vrijgegeven als aan alle volgende voorwaarden wordt voldaan:

1 - het verzoek komt op of voor T2;
2 - de lopende cyclus is niet de lage byte van een woord (op een oneven adres);
3 - de lopende cyclus is niet de eerste acknowledge van een interrupt acknowledge routine;
4 - de processor is niet bezig met het uitvoeren van een gesloten instructie.

Indien de lokale bus stationair is op het moment dat het verzoek wordt gedaan, zijn er twee mogelijkheden:

1 - de lokale bus wordt vrijgegeven tijdens de volgende clock;
2 - binnen 3 klokpulsen zal een geheugencyclus beginnen.

De vier regels voor een lopende aktieve geheugencyclus gelden dan, terwijl aan voorwaarde 1 al is voldaan.

$\overline{\text{LOCK}}$: Lock
(Output, 3-state)
De $\overline{\text{LOCK}}$ uitgang geeft aan dat andere busmasters de besturing van de systeembus niet mogen overnemen zolang $\overline{\text{LOCK}}$ aktief-LAAG is. Het $\overline{\text{LOCK}}$ signaal wordt geaktiveerd door de 'LOCK' prefix instructie en blijft aktief tot de volgende opdracht is beëindigd. Deze uitgang is hoog-impedant tijdens 'hold acknowledge'.

QS1, QS0: Queue Status
(Output)
De status van de wachtrij (queue) is geldig tijdens de CLK-cyclus waarne de queue-operatie wordt verricht. QS1 en QS0 leveren statusinformatie die extern volgen van de interne 8086 instructie wachtrij mogelijk maakt (zie tabel 7/3.1-5).

QS_1	QS_0	Characteristics
0 (LOW)	0	No Operation
0	1	First Byte of Op Code from Queue
1 (HIGH)	0	Empty the Queue
1	1	Subsequent Byte from Queue

Tabel 7/3.1-5: Statusbits voor het extern volgen van de interne wachtrij (queue).

Minimum mode pennen
De volgende functiebeschrijvingen gelden voor de 8086 in de minimum mode (als MN/$\overline{\text{MX}}$ = Vcc).

M/$\overline{\text{IO}}$: Status
(Output, 3-state)
Deze statuslijn is logisch gelijkwaardig aan S2 in de maximum mode en wordt gebruikt om onderscheid te maken tussen een geheugen- en een I/O-toegang. M/$\overline{\text{IO}}$ wordt geldig tijdens de T4 die aan een buscyclus vooraf gaat en blijft geldig tot de laatste T4 van de cyclus (M = HOOG, IO = LAAG). M/$\overline{\text{IO}}$ is hoog-impedant tijdens een lokale bus 'hold acknowledge'.

$\overline{\text{WR}}$: Write
(Output, 3-state)
De write-strobe geeft aan dat de processor bezig is met een schrijfcyclus naar geheugen of I/O, afhankelijk van het M/$\overline{\text{IO}}$-signaal. $\overline{\text{WR}}$ is aktief tijdens T2, T3 en Tw van elke elke schrijfcyclus. Het is aktief-LAAG en hoog-impedant tijdens lokale bus 'hold acknowledge'.

$\overline{\text{INTA}}$: Interrupt Acknowledge
(Output, 3-state)
$\overline{\text{INTA}}$ wordt gebruikt als lees-strobe bij interrupt acknowledge-cyclussen. Het is aktief-LAAG tijdens T2, T3 en Tw van elke interrupt acknowledge cyclus. $\overline{\text{INTA}}$ is nooit zwevend.

ALE: Address Latch Enable
(Output)
Dit signaal wordt door de processor gegeven om het adres in een adreslatch te plaatsen.

Het is een HOOG puls die aktief is gedurende T1 van elke klokcyclus. ALE is nooit zwevend.

DT/$\overline{\text{R}}$: Data Transmit/Receive
(Output, 3-state)
Dit signaal is nodig wanneer voor een minimum systeem een databus transceiver gebruikt moet worden. Hiermee wordt de richting van de datastroom door de transceiver bepaald. Logisch is DT/$\overline{\text{R}}$ gelijkwaardig aan $\overline{\text{S1}}$ in de maximum mode en de timing ervan is dezelfde als voor M/$\overline{\text{IO}}$ (T = HOOG, R = LAAG). Dit signaal is hoog-impedant bij een lokale 'hold acknowledge'.

$\overline{\text{DEN}}$: Data Enable
(Output, 3-state)
Data enable werkt als output enable voor de transceiver in een minimum systeem waarbij een transceiver wordt toegepast. $\overline{\text{DEN}}$ is aktief-LAAG bij het toegang krijgen tot geheugen en I/O en tijdens $\overline{\text{INTA}}$-cyclussen. Bij een lees- of $\overline{\text{INTA}}$-cyclus is het aktief vanaf het midden van T2 tot het midden van T4, terwijl het bij een schrijfcyclus aktief is vanaf het begin van T2 tot het midden van T4. $\overline{\text{DEN}}$ is hoog-impedant tijdens een lokale bus 'hold acknowledge'.

HOLD, HLDA: Hold
(Input, respectievelijk output)
HOLD geeft aan dat een andere master een lokale bus 'hold' vraagt. Om te worden bevestigd, moet HOLD aktief-HOOG worden. De processor die het 'hold' verzoek ontvangt, maakt HLDA als bevestiging HOOG in het midden van een T1 klokcyclus. Tegelijk met de verandering op HLDA zal de processor de lokale bus en de controllijnen hoog-impedant maken. Nadat HOLD LAAG is bevonden, maakt de processor HLDA LAAG en indien de processor nog een extra cyclus moet uitvoeren zal hij opnieuw de lokale bus en controllijnen besturen.
Voor het vrijmaken van de lokale bus gelden dezelfde regels als voor $\overline{\text{RQ}}$/$\overline{\text{GT}}$.
HOLD is geen asynchrone ingang. Indien het systeem de setup tijd niet anders kan garanderen, moet externe synchronisatie plaatsvinden.

Functionele beschrijving

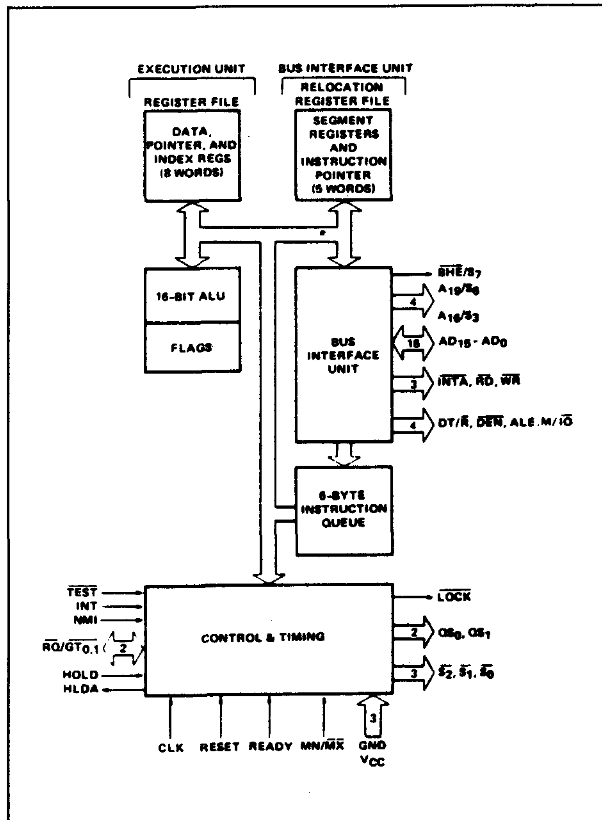
80C86: statisch ontwerp
Alle interne schakelingen van de 80C86 zijn statisch. De interne registers, tellers en latches hebben daarom geen refresh nodig, waardoor de beperking betreffende de minimale bedrijfsfrequentie wegvalt. De 80C86 kan werken op alle frequenties tussen DC en de betreffende maximum frequentie. De processor-klok mag bovendien in de hoge of lage toestand worden gestopt en zo blijven staan. Dit kan bijvoorbeeld nuttig zijn voor foutzoeken in het systeem. Met behulp van de processor-klok kan de 80C86-routine stap voor stap worden doorlopen.

Interne structuur
De interne functies van de 8086 microprocessor kunnen logisch worden gesplitst in twee hoofdgroepen:

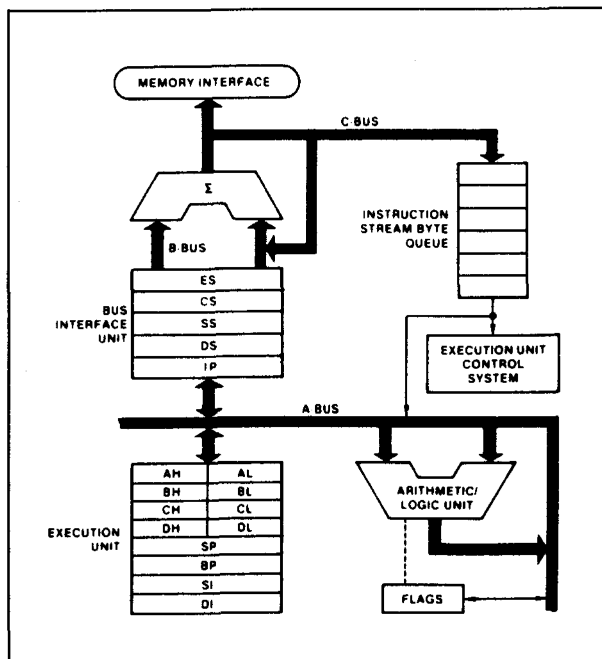
De Bus Interface Unit (BIU) en de Executie/Control Unit (EU), zoals in figuur 7/3.1-3a te zien is. In figuur 7/3.1-3b is dit blokschema nog verder vereenvoudigd, waardoor vergelijking met de 8088 mogelijk wordt. De BIU en de EU werken wel direct samen, maar vervullen hun eigen speciale functies onafhankelijk van elkaar.

De BIU verzorgt de functies die betrekking hebben op het ophalen van de instructies, de wachtrij (queue), ophalen en terugbrengen van operands en adresverplaatsing. Deze eenheid regelt ook de basis bus-besturing. Doordat de instructies worden opgehaald voordat de EU ze nodig heeft (pre-fetch) wordt de bus-bandbreedte vergroot (de processor-snelheid is hiervan in belangrijke mate afhankelijk). De instructies kunnen in een wachtrij van maximaal 6 bytes worden opgeslagen voordat ze worden gedecodeerd en uitgevoerd.

3.1 8086



Figuur 7/3.1-3a: Functioneel blokschema.

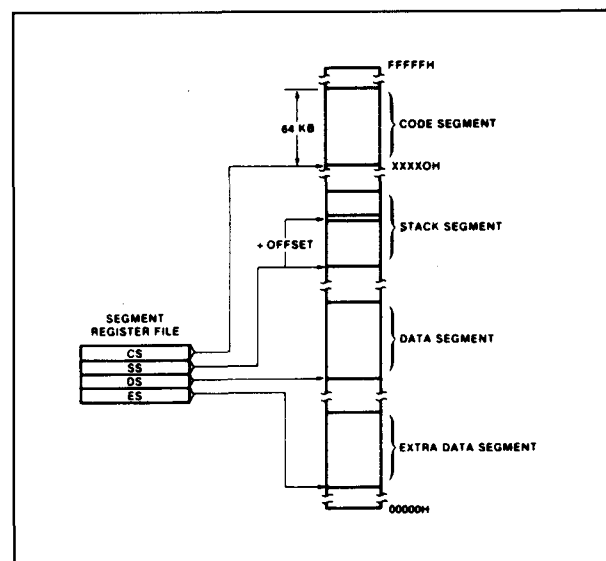


Figuur 7/3.1-3b: Functioneel blokschema.

Door de wachtrij van instructies kan de BIU een zeer efficiënt gebruik maken van het geheugen. Telkens wanneer in de wachtrij ruimte is voor minstens 2 bytes, zal de BIU proberen een ‘word fetch’ geheugencyclus uit te voeren. Hierdoor wordt de dode tijd op de geheugenbus aanzienlijk verminderd. De wachtrij werkt als First-In-First-Out (FIFO) buffer, waaruit de EU de instructies haalt wanneer dat nodig is. Is de wachtrij leeg (bijvoorbeeld na een branch-instructie), dan komt het eerstvolgende byte onmiddellijk voor de EU ter beschikking. De EU ontvangt de instructies uit de BIU wachtrij en levert niet-verplaatste operand-adressen aan de BIU. Geheugen-operands komen via de BIU naar de EU om verwerkt te worden, waarna de resultaten naar de BIU gaan voor opslag.

Geheugen-organisatie

De processor voorziet elke byte geheugen van een 20-bit adres. Het geheugen is georganiseerd als een lineair array van 1 miljoen bytes, geadresseerd tussen 00000(H) en FFFFF(H) (H = hexadecimaal). Het geheugen is logisch onderverdeeld in code, data, extra data en stack-segmenten tot maximaal 64 kbytes per stuk, waarbij elk segment op een 16-bit grens begint (zie figuur 7/3.1-4).



Figuur 7/3.1-4: Geheugen-organisatie

3.1 8086

Alle geheugenplaatsen zijn gerelateerd aan basis-adressen die in high-speed segmentregisters staan. De keuze van de segmenttypen is gebaseerd op de adresseringsbehoeften van de programma's. Moet een segmentregister geselecteerd worden, dan wordt automatisch het type gekozen dat voldoet aan de voor de operatie geldende regels. Alle informatie in een bepaald segmenttype heeft dezelfde logische attributen (bijvoorbeeld code of data). Door het geheugen op te bouwen uit verplaatsbare gebieden met gelijke karakteristieken en de selectie van segmentregisters automatisch te laten plaatsvinden, worden de programma's korter, sneller en beter gestructureerd (zie tabel 7/3.1-6).

TYPE OF MEMORY REFERENCE	DEFAULT SEGMENT BASE	ALTERNATE SEGMENT BASE	OFFSET
Instruction Fetch	CS	NONE	IP
Stack Operation	SS	NONE	SP
Variable (except following)	DS	CS, ES, SS	Effective Address
String Source	DS	CS, ES, SS	SI
String Destination	ES	NONE	DI
BP Used As Base Register	SS	CS, DS, ES	Effective Address

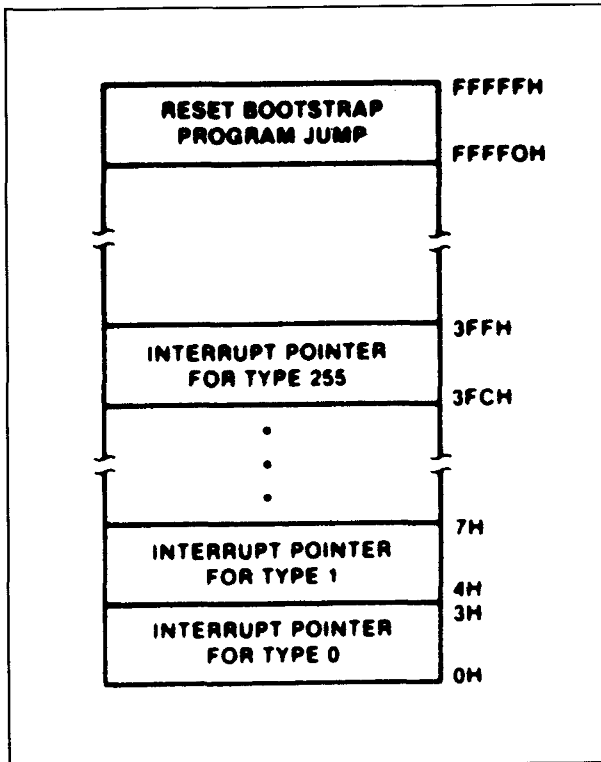
Tabel 7/3.1-6: Geheugenstructuur (indeling met overeenkomstige karakteristieken) voor snellere, kortere en meer gestructureerde programma's.

Woord-operands (16 bit) kunnen op even of oneven adresgrenzen worden neergezet en zijn zodoende niet gebonden aan even adresgrenzen, zoals dat bij vele andere 16-bit computers het geval is. Van adres- en data-operands wordt het minst belangrijke byte van het woord (LSB) opgeborgen in de laagste adreslokatie en het meest belangrijke byte (MSB) in het aansluitend hogere adres. De BIU voert automatisch het juiste aantal geheugentoegangen uit: één indien de woord-operand zich op een even bytegrens bevindt en twee indien de byte-grens oneven is. Hoewel dit ten koste gaat van het prestatievermogen (alleen bij woord-operands, niet bij het ophalen van instructies), is de dubbele toegang transparant voor de software.

Technisch is het geheugen georganiseerd als een hoge bank (D15 tot en met D8) en een lage bank (D7 tot en met D0) van 512 k 8-bit adressen die door de adreslijnen van de processor parallel komen te staan.
Byte data met even adressen wordt over de D7 tot en met D0 buslijnen overgebracht, terwijl oneven geadresseerde data op de buslijnen D15 tot en met D8 terechtkomen. De processor geeft twee enable signalen $\overline{BHE}$ en A0 om selectief te kunnen lezen of schrijven op een even of oneven byte-lokatie of op beide. De instructiestroom wordt in de vorm van woorden uit het geheugen gehaald en indien nodig intern door de processor op byte-niveau geadresseerd.
Voor woord-data heeft de BIU één of twee geheugencyclussen nodig, afhankelijk of het begin-byte van het woord op een even of oneven adres staat. Het werken met woord-operands kan zodoende worden geoptimaliseerd door data op even adresgrenzen te plaatsen. Dit geldt vooral bij gebruik van de stack, aangezien oneven adresverwijzingen naar de stack de context-schakeltijd ongunstig beïnvloeden, bij het verwerken van interrupts of meerdere taken tegelijk.

Sommige plaatsen in het geheugen zijn gereserveerd voor speciale CPU-operaties (zie figuur 7/3.1-5). De plaatsen tussen FFFFOH en FFFFFH zijn bestemd voor bewerkingen die een jump naar de initiële programma laadroutine bevatten. Na een RESET zal de CPU altijd beginnen op adres FFFFOH, zodat daar de jump moet zijn geplaatst. De plaatsen 00000H tot en met 003FFH zijn gereserveerd voor interrupt operaties. Door middel van vier-byte pointers, bestaande uit een 16-bit segmentadres en een 16-bit offset-adres gaat het programma direct door naar een van de 256 mogelijke interrupt serviceroutines. Van de pointer-elementen wordt aangenomen dat zij op hun respectievelijke plaatsen in het gereserveerde geheu-

3.1 8086



Figuur 7/3.1-5: Gereserveerde geheugenplaatsen.

gen waren opgeborgen, voordat de interruptie plaatsvond.

Minimum en maximum modes

De vereiste voorzieningen voor minimum en maximum 8086 systemen zijn zozeer verschillend van elkaar dat het niet mogelijk is om beide efficiënt te bedienen met 40 uniek gedefinieerde aansluitpennen. De 8086 is dan ook voorzien van een 'strap'-pen (MN/$\overline{\text{MX}}$) die de systeemconfiguratie bepaalt. De definitie van een deel van de pennen verandert afhankelijk van de toestand van de strap-pen. Ligt de MN/$\overline{\text{MX}}$-pen aan GND dan kiest de 8086 voor de pennen 24 t/m 31 de maximum mode. Een 8288 buscontroller interpreteert de status-informatie die op de statuslijnen $\overline{\text{S0}}$, $\overline{\text{S1}}$ en $\overline{\text{S2}}$ staat en levert alle timing- en control-signalen die voor de MULTIBUS architectuur nodig is.
Is de MN/$\overline{\text{MX}}$-pen verbonden met Vcc dan genereert de 8086 zelf bus-control signalen op de pennen 24 t/m 31.

Bus bewerkingen

De 8086 heeft een gecombineerde adres- en data-bus die meestal een tijd-gemultiplexte bus wordt genoemd. Deze techniek maakt het meest effectieve gebruik van de pennen mogelijk, waardoor een standaard 40-pens behuizing toegepast kan worden. Deze 'lokale bus' kan direct worden gebufferd en door het gehele systeem gebruikt wanneer de adressen op geheugen en I/O-modulen gelatcht worden. Bovendien kan het demultiplexen van de bus ook bij de processor plaatsvinden met een enkele set 8282 adreslatches indien voor het systeem een niet-gemultiplexte bus gewenst is.

Iedere processor buscyclus bestaat uit minstens vier CLK cyclussen. Deze zijn T1, T2, T3 en T4 (zie figuur 7/3.1-2). Het adres wordt gedurende T1 door de processor uitgezonden en data wordt gedurende T3 en T4 op de bus overgedragen. T2 wordt voornamelijk gebruikt om de richting van de bus te veranderen tijdens lees-operaties. In het geval dat de geadresseerde schakeling een 'NOT READY' indicatie geeft, worden wachttijden (Tw) tussen T3 en T4 geplaatst. Elke tussengevoegde wachttijd duurt even lang als een CLK-cyclus. Tussen de door de 8086 bestuurde buscyclussen kunnen perioden optreden die stationaire toestanden, 'idle states' (Ti) of niet-aktieve CLK-cyclussen worden genoemd. De processor gebruikt deze cyclussen voor interne huishoudelijke zaken.

Gedurende T1 van elke buscyclus wordt het ALE-signaal (Address Latch Enable) uitgezonden door de processor of door de 8288 buscontroller, afhankelijk van MN/$\overline{\text{MX}}$. Op de achterflank van deze puls kunnen een geldig adres en bepaalde statusinformatie voor de cyclus worden gelatcht.

In de maximum mode worden de statusbits $\overline{\text{S0}}$, $\overline{\text{S1}}$ en $\overline{\text{S2}}$ door de buscontroller gebruikt om de soort bushandeling volgens tabel 7/3.1-4 te identificeren.

3.1 8086

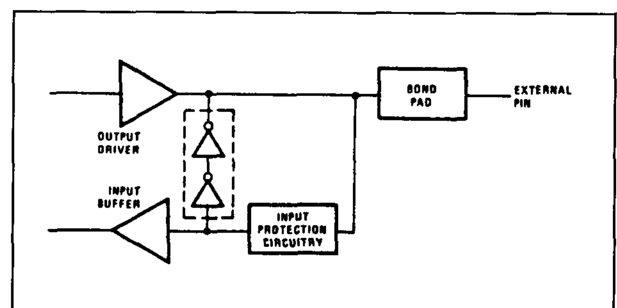
De statusbits S3 tot en met S7 worden tijdgemultiplext met hoge orde adresbits en het $\overline{BHE}$ signaal, waardoor zij geldig zijn van T2 tot en met T4. Met S3 en S4 wordt aangegeven welk segmentregister bij deze buscyclus werd gebruikt voor de vorming van het adres (zie ook tabel 7/3.1-2). S5 is een afspiegeling van het PSW interrupt enable bit. S6 is niet gedefinieerd en S7 is een reserve statusbit.

I/O adressering
In de 8086 kunnen I/O handelingen maximaal 64 k I/O byteregisters op 32 k I/O woordregisters adresseren. Het I/O adres verschijnt in hetzelfde formaat als het geheugenadres op de buslijnen A15 tot en met A0. De adreslijnen A19 tot en met A16 zijn nul bij I/O operaties. De variabele I/O instructies die het DX-register als pointer gebruiken kunnen alle adressen bereiken, terwijl de directe I/O instructies rechtstreeks een of twee van de 256 I/O byte lokaties in pagina 0 van de I/O adresruimte adresseren. I/O poorten worden op dezelfde manier geadresseerd als geheugenplaatsen. Even adresbytes worden verzonden over de buslijnen D7 tot en met D0 en oneven adresbytes over D15 tot en met D8. Let op dat elk register binnen een 8-bit randschakeling die op het laagste deel van de bus is geplaatst, even geadresseerd moet zijn.

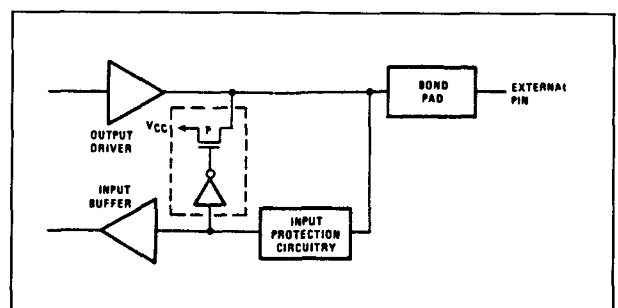
Externe Interface

Resetten van de processor en initialisatie
De processor wordt geïnitialiseerd en start-up vindt plaats door de RESET-pen te activeren (HOOG). De RESET van de 8086 moet langer dan vier CLK-cyclussen HOOG zijn. De 8086 stopt de bewerkingen op het HOOG gaan van RESET en blijft dan slapend zolang RESET HOOG is. Bij het LAAG gaan van RESET wordt een interne reset-volgorde getriggerd. Na deze onderbreking werkt de 8086 normaal, beginnend met de instructie in het absolute adres FFFF0H (zie ook figuur 7/3.1-5). De RESET ingang wordt intern gesynchroniseerd op de processor-klok. Bij het initialiseren mag de HOOG-naar-LAAG overgang van RESET niet eerder dan 50 μs (of 4 klokcyclussen als deze langer duren) na power-up plaatsvinden om de 8086 de gelegenheid te geven volledig geïnitialiseerd te worden. Er mag geen NMI komen voor de tweede CLK-cyclus na het einde van RESET.

80C86: Bus Hold-voorzieningen
Om situaties met grote stroomsterkten (als gevolg van zwevende ingangen van CMOS schakelingen) te vermijden, zijn op de pennen 2 tot en met 16, 26 tot en met 32 en 34 tot en met 39 van de 80C86 zogenaamde 'bus hold' voorzieningen aangebracht (zie figuur 7/3.1-6a en b). Deze schakelingen houden de laatste logische toestand vast indien er geen aandrijvende bron is (dat wil zeggen een niet-verbonden pen of een driver die in de hoog-impedante toestand gaat). Om de bus hold schakelingen te oversturen moet de externe driver in staat zijn om sink- of source-



Figuur 7/3.1-6a: Bus Hold schakeling (pennen 2 tot en met 16, 34 tot en met 39;



Figuur 7/3.1-6b: Bus Hold schakeling (pennen 26 tot en met 32).

3.1 8086

stromen van ongeveer 400 μA bij geschikte ingangsspanningen te leveren. Aangezien deze bus hold schakelingen aktief en niet 'resistief' zijn, is de hiervoor benodigde voedingsstroom verwaarloosbaar, terwijl het gedissipeerde vermogen ten opzichte van optrekweerstanden aanzienlijk lager is.

Interrupt handelingen
Interrupt-operaties kunnen twee bronnen hebben: veroorzaakt door software of door hardware. De interrupties als gevolg van software en de software-aspecten van hardware-interrupties worden gespecificeerd in de beschrijving van de instructieset. Hardware interrupties kunnen worden geklassificeerd als wel- of niet-maskeerbaar.

Een interruptie resulteert in het overbrengen van de besturing naar een nieuwe programma-lokatie. Voor dit doel is op de absolute adressen tussen 0 en 3FFH een uit 256 elementen bestaande tabel gereserveerd die adres-pointers bevat die verwijzen naar de interrupt service program lokaties. Elk element in de tabel is vier bytes groot en komt overeen met een interruptie-'type'. Een interrumperende schakeling levert tijdens het bevestigen van de interruptie (interrupt acknowledge) een 8-bit typenummer dat wordt gebruikt om via het juiste element naar de nieuwe lokatie voor het uitvoeren van de interrupt-routine te gaan. Alle vlaggen en zowel het Code Segment register als het Instruction Pointer register worden ge'saved' door de INTA volgorde. Deze worden bij de uitvoering van een Interrupt Return (IRET) instructie weer teruggebracht.

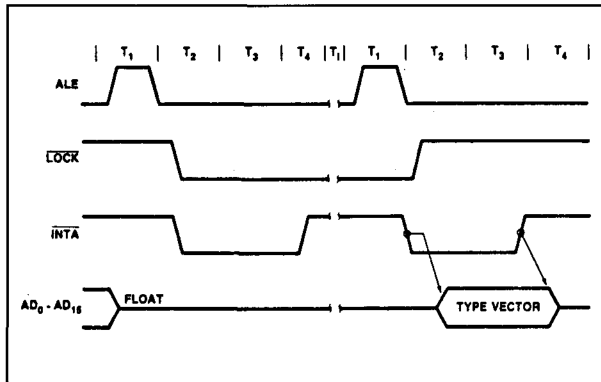
Niet-maskeerbare interruptie (NMI)
De processor heeft één pen voor niet-maskeerbare interrupties (NMI) die een hogere prioriteit heeft dan de maskeerbare interrupt-request pen (INTR). Deze is bijvoorbeeld zeer geschikt voor het aktiveren van een routine bij het wegvallen van de voeding. De NMI wordt getriggerd door een hooggaande flank.

NMI moet langer dan twee CLK-cyclussen HOOG zijn maar behoeft niet te worden gesynchroniseerd op de klok. Elke LAAG-naar-HOOG-overgang van NMI wordt op de chip gelatcht en wordt òf aan het einde van de lopende instructie bediend òf tussen twee hele verplaatsingen van een bloktype instructie. Worst-case responsie op NMI kan optreden bij vermenigvuldigen, delen en variabele verschuif-instructies. Er is geen voorschrift wanneer de neergaande flank moet verschijnen; deze mag voor, tijdens of na de bediening van NMI komen. Een volgende hooggaande flank triggert een volgende reactie indien deze optreedt na het begin van de NMI-procedure. Het signaal mag geen spikes bevatten en mag op de neergaande flank niet natrillen.

Maskeerbare interrupties (INTR)
De 8086 heeft één interrupt request ingang (INTR) die intern met software gemaskeerd kan worden door het resetten van het interrupt enable VLAG-statusbit. Het interrupt request signaal is niveau-getriggerd en wordt intern gesynchroniseerd op de opgaande flank van elke CLK-cyclus. Er wordt gereageerd op INTR wanneer het aanwezig (HOOG) is tijdens de klokperiode voorafgaande aan het einde van de lopende instructie of het einde van een gehele verplaatsing (move) voor een blok-type instructie. Tijdens de behandeling van de interrupt routine worden verdere onderbrekingen geblokkeerd. Het enable bit wordt gereset als deel van de reactie op een onderbreking (INTR, NMI, software interrupt of single step) hoewel het vlagregister, dat automatisch op de stack wordt gezet, de toestand van de processor voorafgaande aan de interruptie weergeeft. Totdat het oude vlagregister is teruggebracht blijft het enable bit nul, tenzij het door een opdracht speciaal wordt geset.

Bij de behandeling van een interruptie (zie figuur 7/3.1-7) voert de processor twee opeenvolgende bevestigingscyclussen (interrupt acknowledge) uit. De 8086 zendt het LOCK

3.1 8086



Figuur 7/3.1-7: Bevestiging van een interruptie (interrupt acknowledge sequence).

signaal uit van T2 van de eerste buscyclus tot T2 van de tweede. Een lokaal bus 'hold' verzoek wordt niet gehonoreerd tot het einde van de tweede buscyclus. In de tweede buscyclus wordt een byte door de 8259A Interrupt Controller naar de 8086 gebracht die de herkomst van de interruptie (type) identificeert. Deze byte wordt met vier vermenigvuldigd en als pointer gebruikt in de interruptvector opzoektabel. Een INTR signaal dat HOOG wordt gelaten zal voortdurend worden beantwoord binnen de beperkingen van de enable bit- en sample periode. De INTERRUPT RETURN opdracht bevat een 'vlagpop' waarmee de status van het originele interrupt enable bit wordt teruggezet wanneer de vlaggen weer worden opgeborgen.

HALT
Wanneer een software HALT-instructie wordt uitgevoerd geeft de processor aan dat hij in de halt-toestand gaat op een van de twee manieren, afhankelijk van de gekozen mode. In de minimum mode levert de processor één ALE zonder kwalificerende buscontrolsignalen. In de maximum mode geeft de processor de juiste HALT-status op $\overline{S2}$, $\overline{S1}$ en $\overline{S0}$, terwijl de 8288 buscontroller één ALE levert. De 8086 zal de HALT-toestand niet verlaten als tijdens 'HALT' een lokale bus 'hold' binnenkomt. In dat geval geeft de processor opnieuw de HALT-indicatie. Een NMI of interrupt request (wanneer er interrupts enabled zijn) aan het einde van de bus hold periode of een RESET dwingt de 8086 uit de HALT-toestand.

Read/Modify/Write (semafoor)
Bewerkingen via LOCK
De processor geeft $\overline{LOCK}$ statusinformatie wanneer tijdens het uitvoeren van een opdracht direct aansluitende buscyclussen nodig zijn. De processor is hierdoor in staat om read/modify/write handelingen op het geheugen uit te voeren (via de 'verwissel register met geheugen' instructie) zonder dat een andere systeem busmaster interveniërende geheugencyclussen ontvangt. Dit is nuttig om in multi-processorsystemen 'test en set lock' operaties mogelijk te maken. Het $\overline{LOCK}$ signaal wordt geaktiveerd (LAAG gedwongen) in de klokcyclus volgende op die waarin de software 'LOCK' prefix instructie door de EU wordt gedecodeerd. Op het einde van de laatste buscyclus van de instructie die op de 'LOCK' prefix instructie volgt, wordt het $\overline{LOCK}$ signaal gedeaktiveerd. Terwijl LOCK aktief is wordt een request op een $\overline{RQ}/\overline{GT}$-pen vastgelegd om te worden gehonoreerd aan het einde van de LOCK.

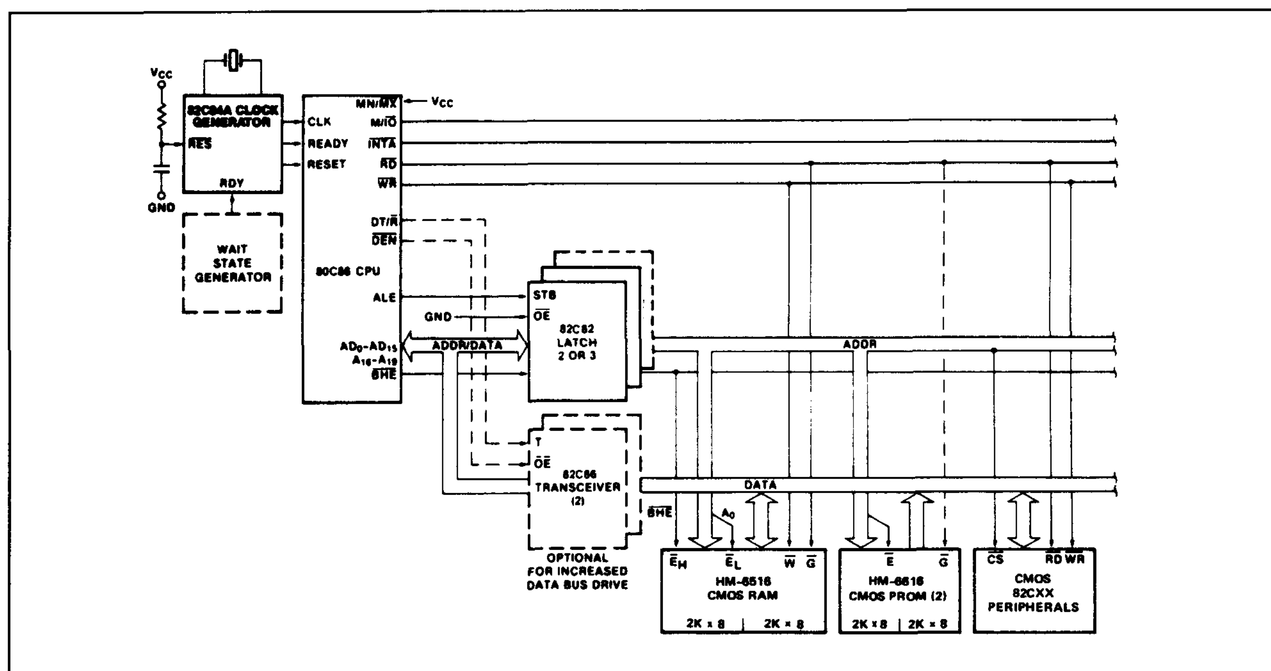
Externe synchronisatie via TEST
Als een alternatief op de interrupties en algemene I/O mogelijkheden heeft de 8086 een $\overline{TEST}$ ingang die met software getest kan worden. Te allen tijde kan het programma een WAIT opdracht uitvoeren. Als in die tijd het $\overline{TEST}$ signaal inaktief (HOOG) is, wordt de uitvoering van het programma uitgesteld waarbij de processor wacht tot $\overline{TEST}$ aktief wordt. Het moet tenminste vijf klokcyclussen aktief blijven. Tot die tijd wordt de WAIT instructie telkens opnieuw uitgevoerd. Deze aktiviteit kost geen buscyclussen. De processor blijft tijdens het wachten in een stationaire toestand. Als een bus HOLD binnenkomt gaan alle 8086 drivers in de hoogimpedante toestand. Als er interrupties enabled zijn, kunnen deze optreden terwijl de processor wacht. Gebeurt dit dan haalt de

3.1 8086

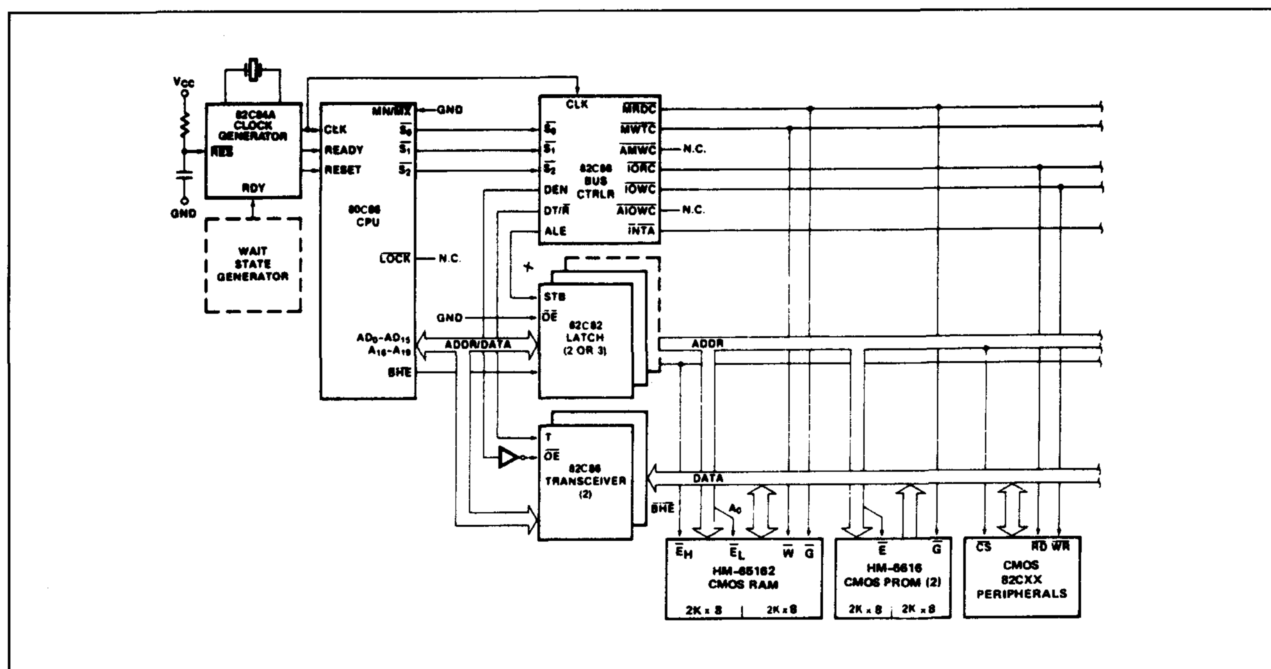
processor de WAIT instructie nog een keer extra op, voert de interrupt routine uit en haalt daarna de WAIT opdracht nogmaals op om hem weer uit te voeren.

Principiële systeem timing

In de figuren 7/3.1-8a en b zijn de principiële systeemconfiguraties te zien, waarbij de processor in de minimum, respectievelijk maxi-



Figuur 7/3.1-8a: Minimum mode configuratie.



Figuur 7/3.1-8b: Maximum mode configuratie.

3.1 8086

mum mode werkt. In de minimum mode is de MN/$\overline{\text{MX}}$-pen aan Vcc gelegd en zendt de processor direct bus-control signalen uit ($\overline{\text{RD}}$, $\overline{\text{WR}}$, enzovoorts). In de maximum mode ligt de MN/$\overline{\text{MX}}$-pen aan GND en zendt de processor gecodeerde status-informatie uit die de 8288 buscontroller gebruikt voor het opwekken van MULTIBUS compatibele busbesturingssignalen.
In figuur 7/3.1-2 is het verband tussen de signalen te zien.

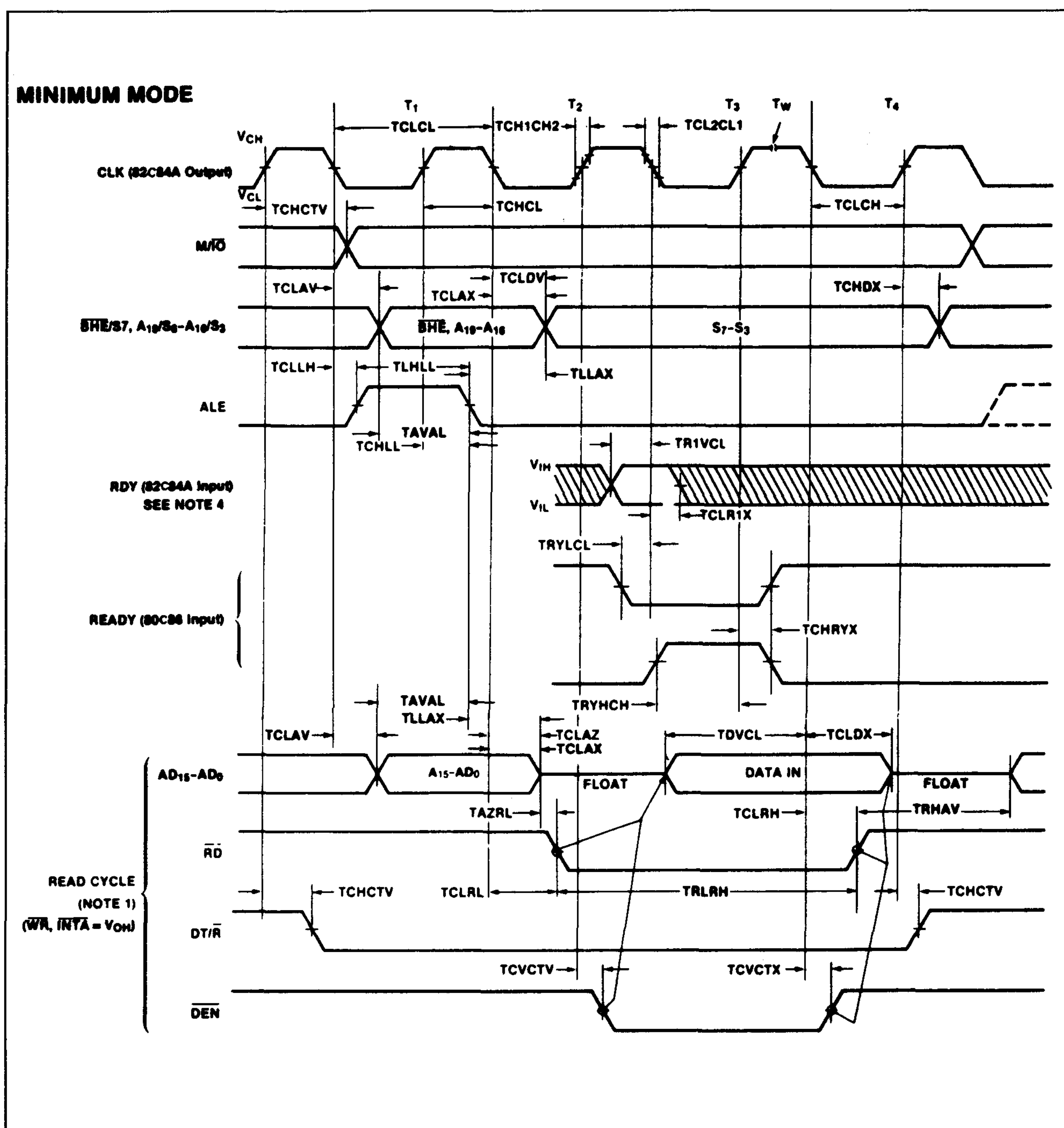
Systeem timing - minimum systeem
De leescyclus begint op T1 met het aanbrengen van het Address Latch Enable (ALE) signaal. De laaggaande achterflank van dit signaal wordt gebruikt om de adres-informatie die op dat moment op de lokale bus beschikbaar is in de 8282/8283 te latchen. De $\overline{\text{BHE}}$ en A0 signalen adresseren het hoge of lage byte of beide bytes. Van T1 tot T4 laat het M/$\overline{\text{IO}}$-signaal een geheugen- of I/O-handeling zien. Bij T2 wordt het adres van de lokale bus verwijderd en gaat de bus in de hoogimpedante toestand. Het read control signaal verschijnt ook op T2. Het leessignaal ($\overline{\text{RD}}$) maakt dat de databus drivers van de geadresseerde schakeling aktief worden op de lokale bus. Enige tijd later zal geldige data op de bus beschikbaar zijn en zet de geadresseerde schakeling de READY-lijn HOOG. Als de processor het leessignaal weer HOOG maakt, zet de geadresseerde schakeling zijn busdrivers weer in de hoogimpedante toestand. Indien een transceiver (8286/8287) nodig is om de lokale 8086 bus te bufferen, levert de 8086 de signalen DT/$\overline{\text{R}}$ en $\overline{\text{DEN}}$.

Ook een schrijfcyclus begint met het opwekken van ALE en de emissie van het adres. Het M/$\overline{\text{IO}}$-signaal wordt weer verstrekt om een geheugen- of I/O-handeling aan te geven. Op T2, onmiddellijk na het verzenden van het adres, stuurt de processor de data naar de geadresseerde plaats. Deze data blijft geldig tot het midden van T4. Gedurende T2, T3 en Tw geeft de processor het write control signaal. Het schrijfsignaal ($\overline{\text{WR}}$) wordt bij het begin van T2 aktief (in tegenstelling tot read, dat iets in T2 is vertraagd om de bus tijd te geven om te zweven).
De $\overline{\text{BHE}}$ en A0 signalen worden gebruikt om de juiste byte(s) van het geheugen/IO woord te selecteren dat gelezen of geschreven moet worden (tabel 7/3.1-3). I/O-poorten worden op dezelfde manier geadresseerd als geheugenplaatsen. Bytes op even adressen worden verstuurd op de D7 tot en met D0 bus en oneven bytes op D15 tot en met D8.

Het belangrijkste verschil tussen de interrupt acknowledge cyclus en een leescyclus is dat het interrupt acknowledge signaal ($\overline{\text{INTA}}$) verschijnt in plaats van het read signaal ($\overline{\text{RD}}$) en dat de adresbus zweeft (zie figuur 7/3.1-7). In de laatste van twee aansluitende $\overline{\text{INTA}}$-cyclussen wordt een byte informatie van de buslijnen D7 tot en met D0 gelezen die door de interrupt systeemlogika (bijvoorbeeld een 8259A PIC) wordt geleverd. Deze byte identificeert de bron (type) van de interruptie. Hij wordt met vier vermenigvuldigd en gebruikt als pointer in een interrupt vector opzoektabel, zoals al eerder werd beschreven.

Bus timing - gemiddeld complexe systemen
Voor systemen met een gemiddelde complexiteit wordt de MN/$\overline{\text{MX}}$-pen aan GND gelegd en wordt het systeem uitgebreid met de 8288 buscontroller, een 8282/8283 latch voor het latchen van het systeemadres en een 8286/8287 transceiver voor busmanipulaties die boven de mogelijkheden van de 8086 uitgaan. ALE, $\overline{\text{DEN}}$ en DT/$\overline{\text{R}}$ worden nu door de 8288 gegenereerd in plaats van door de processor, waarbij de timing vrijwel gelijk blijft. De status-uitgangen $\overline{\text{S2}}$, $\overline{\text{S1}}$ en $\overline{\text{S0}}$ van de 8086 leveren informatie over het type cyclus en worden 8288 ingangen. Deze buscyclus informatie specificeert read (code, data of I/O), write (data of I/O), interrupt acknowledge of software halt. De 8288 levert zodoende besturingssignalen die geheugen

read of write, I/O read of write, interrupt acknowledge of halt specificeren. De 8288 geeft naar wens twee typen write strobe af: normaal of geavanceerd. Bij de normale write strobes is de data geldig op de voorflank van write. De geavanceerde write strobes hebben dezelfde timing als read strobes, waardoor data niet geldig is op de voorflank van write. De 8286/8287 transceiver ontvangt de gebruikelijke T en OE ingangssignalen van de DT/$\overline{R}$ en $\overline{DEN}$ uitgangen van de 8288. De pointer naar de interrupt vectortabel die



Figuur 7/3.1-9a: Bustiming - minimum mode systeem.

3.1 8086

D.C. ELECTRICAL CHARACTERISTICS

VCC = 5.0V±10%; T_A = 0°C to +70°C (C80C86);

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
VIH	Logical One Input Voltage	2.0 2.2		V V	C80C86, I80C86 M80C86
VIL	Logical Zero Input Voltage		0.8	V	
VIHC	CLK Logical One Input Voltage	VCC -0.8V		V	
VILC	CLK Logical Zero Input Voltage		0.8	V	
VOH	Output High Voltage	3.0 VCC -0.4		V V	IOH = -2.5mA IOH = -100μA
VOL	Output Low Voltage		0.4	V	IOL = +2.5mA
IIL	Input Leakage Current	-1.0	1.0	μA	OV ≤ VIN ≤ VCC
IBHH	Input Leakage Current-Bus Hold High	-50	-400	μA	VIN = 3.0V (see Note 1)
IBHL	Input Leakage Current-Bus Hold Low	50	400	μA	VIN = 0.8V (see Note 2)
IO	Output Leakage Current	-10.0	10.0	μA	OV ≤ VO ≤ VCC
ICCSB	Standby Power Supply Current		500	μA	VCC = 5.5V VIN = VCC or GND Outputs unloaded
ICCOP	Operating Power Supply Current		10	mA/MHz	TA = 25°C VCC = 5V, TYPICAL

CAPACITANCE

TA = 25°C; VCC = GND = OV; VIN = +5V or GND

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
CIN*	Input Capacitance		5	pf	FREQ = 1MHz Unmeasured pins returned to GND
COUT* CI/O*	Output Capacitance I/O Capacitance		15 20	pf pf	

* Guaranteed and sampled, but not 100% tested

Note 1: IBHH should be measured after raising VIN to VCC and then lowering to 3.0V on the following pins: 2-16, 26-32, 34-39.

Note 2: IBHL should be measured after lowering VIN to GND and then raising to 0.8V on the following pins: 2-16, 26-32, 34-39.

Tabel 7/3.1-7: Gelijkstroom-eigenschappen van de 80C86.

3.1 8086 A.C. CHARACTERISTICS

VCC = +5V±10%, GND = 0V : TA = 0°C to +70°C

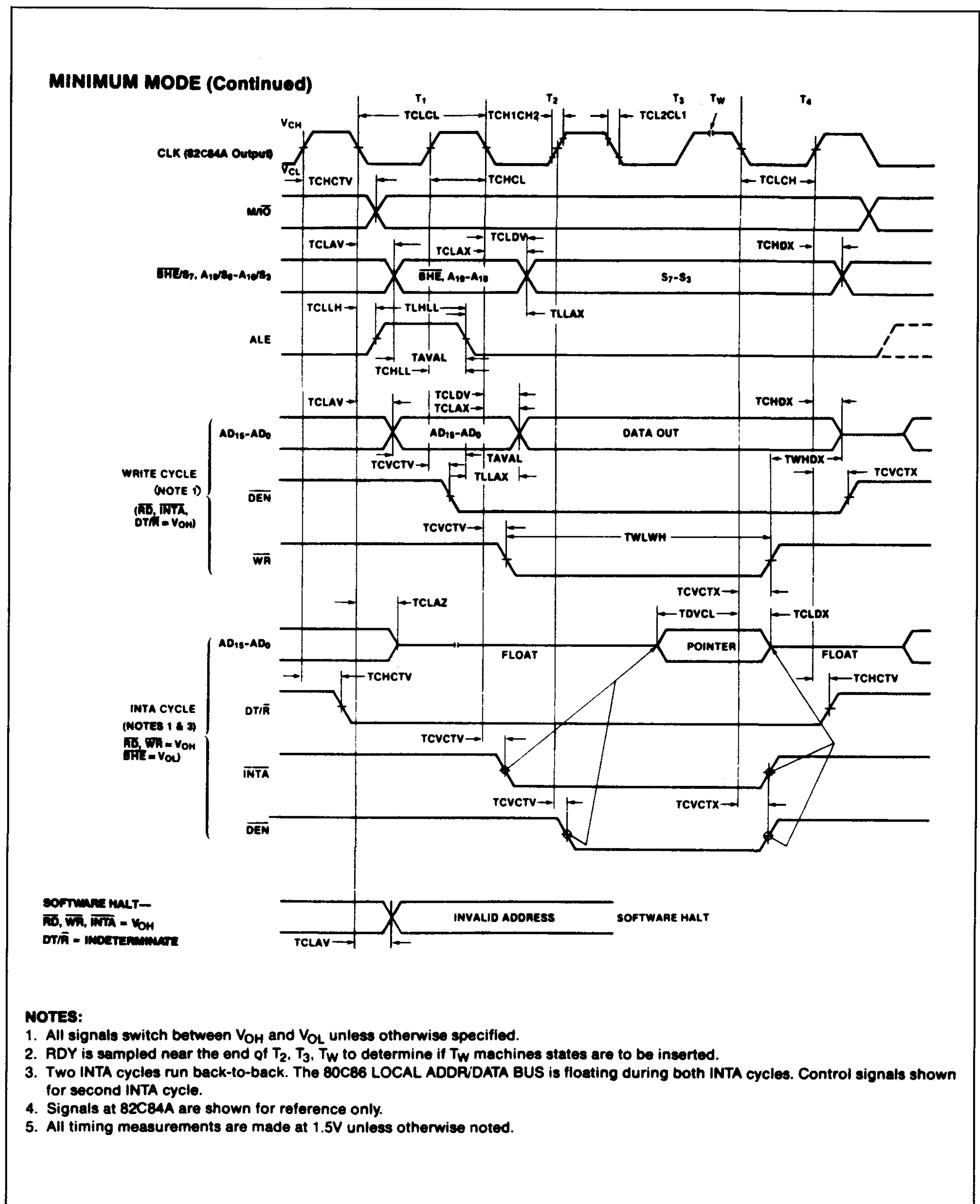
MINIMUM COMPLEXITY SYSTEM

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
TIMING REQUIREMENTS					
TCLCL	CLK Cycle Period	200		ns	
TCLCH	CLK Low Time	118		ns	
TCHCL	CLK High Time	69		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		ns	CL = 20 -100pF
TCLDX	Data in Hold Time	10		ns	
TR1VCL	RDY Setup Time into 82C84A (see Note 1,2)	35		ns	
TCLR1X	RDY Hold Time into 82C84A (see Note 1,2)	0			
TRYHCH	READY Setup Time into 80C86	118		ns	
TCHRYX	READY Hold Time into 80C86	30		ns	
TRYLCL	READY Inactive to CLK (see Note 3)	-8		ns	
THVCH	HOLD Setup Time	35		ns	
TINVCH	INTR, NMI, TEST Setup Time (See Note 2)	30		ns	
TILIH	Input Rise Time (Except CLK)		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		15	ns	From 2.0V to 0.8V
TIMING RESPONSES					
TCLAV	Address Valid Delay	10	110	ns	CL = 20 -100pF
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	80	ns	
TLHLL	ALE Width	TCLCH-20		ns	
TCLLH	ALE Active Delay		80	ns	
TCHLL	ALE Inactive Delay		85	ns	
TLLAX	Address Hold Time to ALE Inactive	TCHCL-10		ns	
TCLDV	Data Valid Delay	10	110	ns	
TCHDX	Data Hold Time	10		ns	
TWHDX	Data Hold Time After WR	TCLCH-30		ns	
TCVCTV	Control Active Delay 1	10	110	ns	
TCHCTV	Control Active Delay 2	10	110	ns	
TCVCTX	Control Inactive Delay	10	110	ns	
TAZRL	Address Float to READ Active	0		ns	
TCLRL	RD Active Delay	10	165	ns	
TCLRH	RD Inactive Delay	10	150	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL-45		ns	
TCLHAV	HLDA Valid Delay	10	160	ns	
TRLRH	RD Width	2TCLCL-75		ns	
TWLWH	WR Width	2TCLCL-60		ns	
TAVAL	Address Valid to ALE Low	TCLCH-60		ns	
TOLOH	Output Rise Time		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		15	ns	From 2.0V to 0.8V

NOTES: 1. Signal at 82C84A shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T2 state (8 ns into T3).

Tabel 7/3.1-8: Wisselstroom-eigenschappen (timing van de 80C86 in de minimum mode).

3.1 8086



NOTES:

1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Two INTA cycles run back-to-back. The 80C86 LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control signals shown for second INTA cycle.
4. Signals at 82C84A are shown for reference only.
5. All timing measurements are made at 1.5V unless otherwise noted.

Figuur 7/3.1-9b: Bustiming - minimum mode systeem (vervolg).

3.1 8086

MAX MODE SYSTEM (USING 82C88 BUS CONTROLLER)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
TIMING REQUIREMENTS					
TCHCL	CLK Cycle Period	200		ns	
TCLCH	CLK Low Time	118		ns	
TCHCL	CLK High Time	69		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data in Setup Time	30		ns	
TCLDX	Data in Hold Time	10		ns	
TR1VCL	RDY Setup Time into 82C84A (see Notes 1,2)	35		ns	
TCLR1X	RDY Hold Time into 82C84A (see Notes 1,2)	0		ns	
TRYHCH	READY Setup Time into 80C86	118		ns	
TCHRYX	READY Hold Time into 80C86	30		ns	CL = 20 - 100 pF
TRYLCL	READY inactive to CLK (see Note 4)	-8		ns	
TINVCH	Setup Time for Recognition (INTR, NMI, TEST) (see Note 2)	30		ns	
TGVCH	RQ/GT Setup Time	30		ns	
TCHGX	RQ Hold Time into 80C86	40		ns	
TILIH	Input Rise Time (Except CLK)		15	ns	From 0.8V to 2.0V
TIHIL	Input Fall Time (Except CLK)		15	ns	From 2.0V to 0.8V
TIMING RESPONSES					
TCLML	Command Active Delay (see Note 1)	5	35	ns	
TCLMH	Command Inactive	5	35	ns	
TRYHSH	READY Active to Status Passive (see Note 3)		110	ns	
TCHSV	Status Active Delay	10	110	ns	
TCLSH	Status Inactive Delay	10	130	ns	
TCLAV	Address Valid Delay	10	110	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	80	ns	
TSVLH	Status Valid to ALE High (see Note 1)		20	ns	
TSVMCH	Status Valid to MCE High (see Note 1)		30	ns	CL = 20 - 100 pf for all 80C86 Outputs (In addition to 80C86 self-load)
TCLLH	CLK Low to ALE Valid		15	ns	
TCLMCH	CLK Low to MCE High (see Note 1)		25	ns	
TCHLL	ALE Inactive Delay (see Note 1)	4	18	ns	
TCLMCL	MCE Inactive Delay (see Note 1)		15	ns	
TCLDV	Data Valid Delay	10	110	ns	
TCHDX	Data Hold Time	10		ns	
TCVNV	Control Active Delay (see Note 1)	5	45	ns	
TCVNX	Control Inactive Delay (see Note 1)	10	45	ns	
TAZRL	Address Float to Read Active	0		ns	
TCLRL	RD Active Delay	10	165	ns	
TCLRH	RD Inactive Delay	10	150	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL-45		ns	
TCHDTL	Direction Control Active Delay (see Note 1)		50	ns	
TCHDTH	Direction Control Inactive Delay (see Note 1)		30	ns	
TCLGL	GT Active Delay	0	85	ns	
TCLGH	GT Inactive Delay	0	85	ns	
TRLRH	RD Width	2TCLCL-75		ns	
TOLOH	Output Rise Time		15	ns	From 0.8V to 2.0V
TOHOL	Output Fall Time		15	ns	From 2.0V to 0.8V

NOTES: 1. Signal at 82C84A or 82C88 shown for reference only.
2. Setup requirement for asynchronous signal only to guarantee recognition at next CLK.
3. Applies only to T3 and wait states.
4. Applies only to T2 state (8 ns into T3).

Tabel 7/3.1-9: Timing van de 80C86 in de maximum mode.

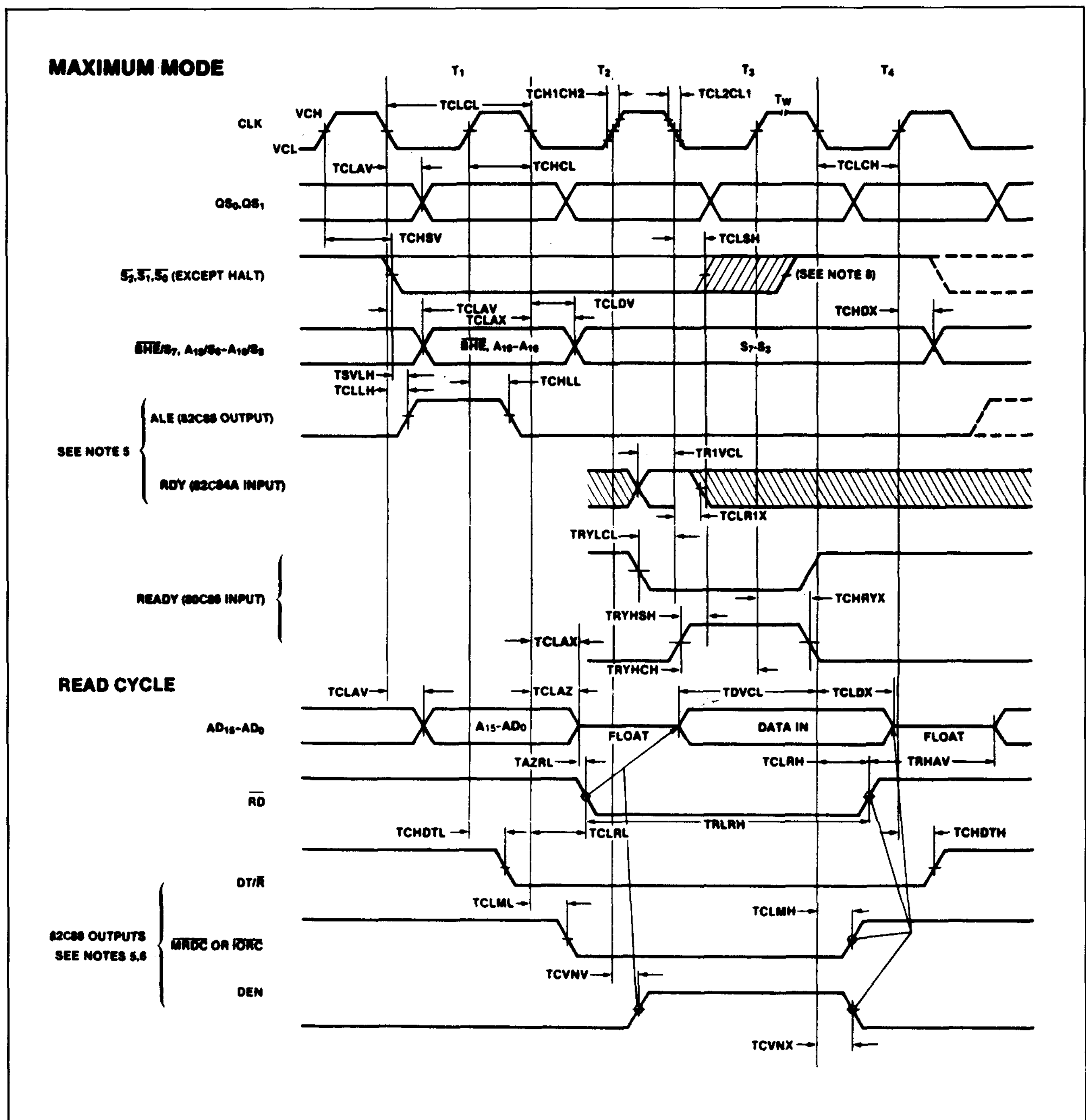
3.1 8086

tijdens de tweede INTA cyclus wordt aangegeven kan afkomstig zijn van een 8259A op de lokale bus of de systeembus. Als de master 8259A Priority Interrupt Controller op de lokale bus is aangesloten, is een TTL-poort nodig om de 8286/8287 transceiver te blokkeren wanneer tijdens de interrupt acknowledge en software 'poll' uit de master 8259A wordt gelezen.

Specificaties

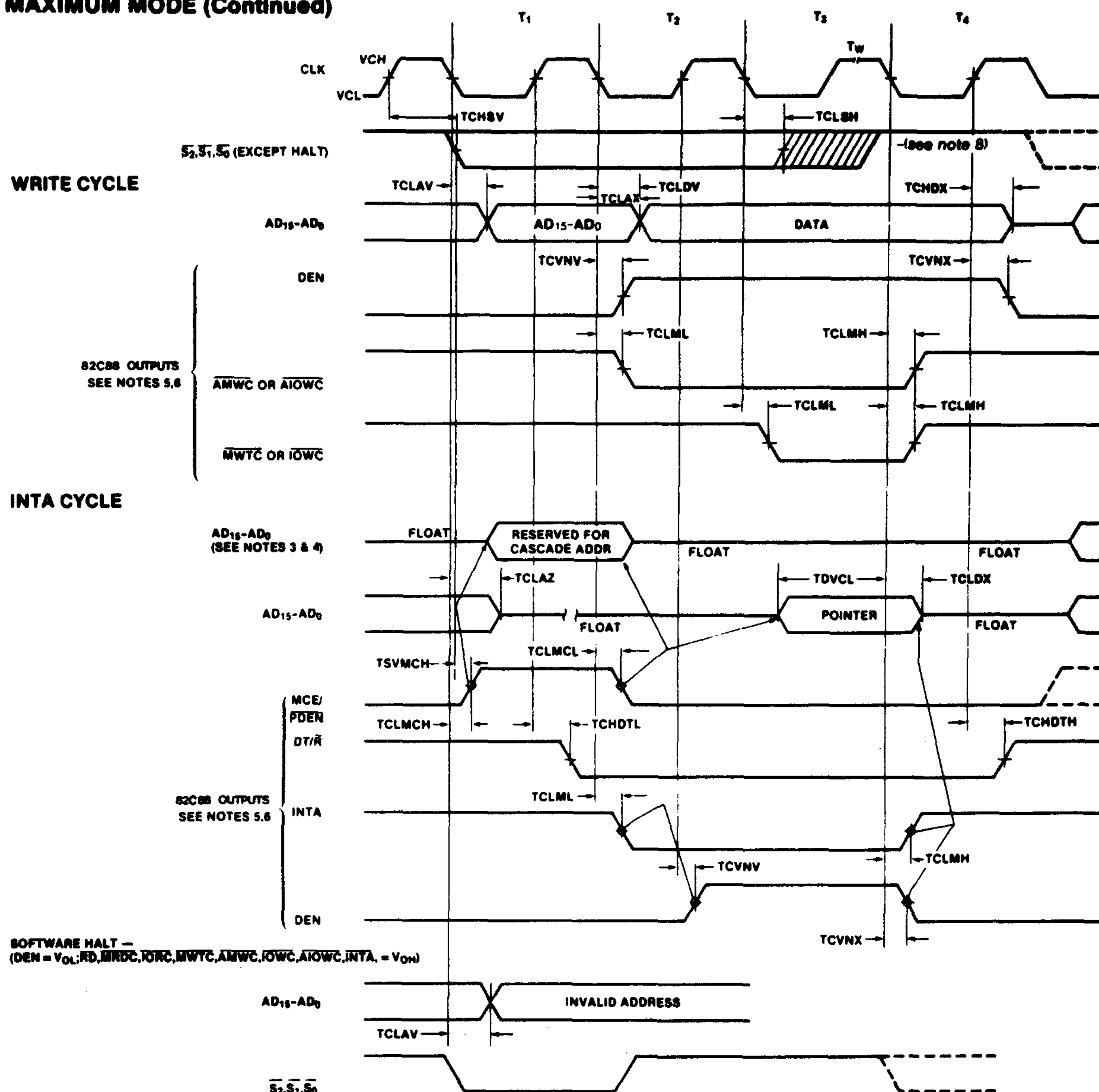
Absolute maximum waarden

Bedrijfstemperatuur: 0 tot 70 °C



Figuur 7/3.1-10a: Bustiming - maximum mode systeem (met gebruik van een 8288).

3.1 8086

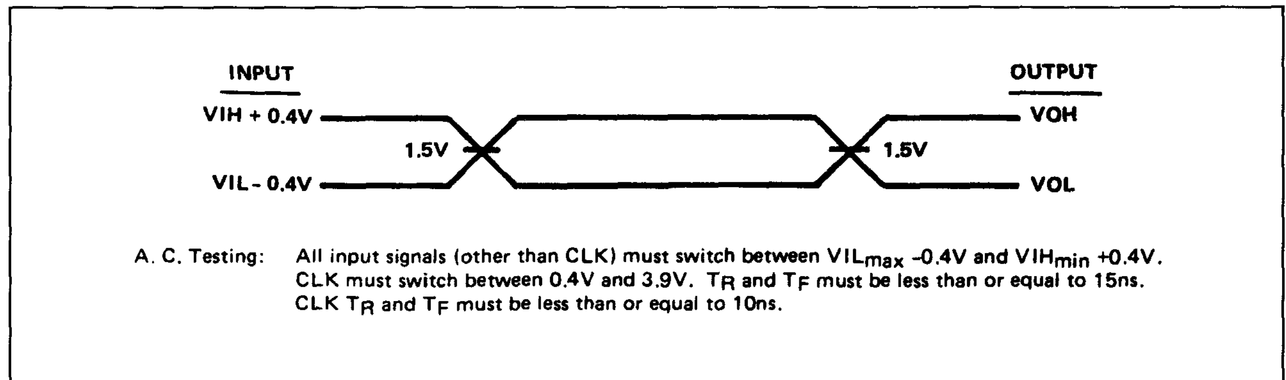


NOTES:

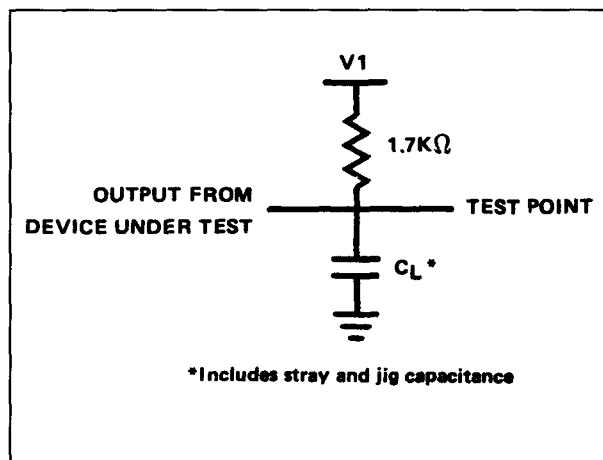
1. All signals switch between V_{OH} and V_{OL} unless otherwise specified.
2. RDY is sampled near the end of T_2, T_3, T_W to determine if T_W machines states are to be inserted.
3. Cascade address is valid between first and second INTA cycle.
4. Two INTA cycles run back-to-back. The 80C86 LOCAL ADDR/DATA BUS is floating during both INTA cycles. Control for pointer address is shown for second INTA cycle.
5. Signals at 82C84A or 82C88 are shown for reference only.
6. The issuance of the 82C88 command and control signals (MRDC, MWTC, AMWC, IORC, IOWC, AIOWC, INTA and DEN) lags the active high 82C88 CEN.
7. All timing measurements are made at 1.5V unless otherwise noted.
8. Status inactive in state just prior to T_4.

Figuur 7/3.1-10b: Bustiming - maximum mode systeem (vervolg).

3.1 8086



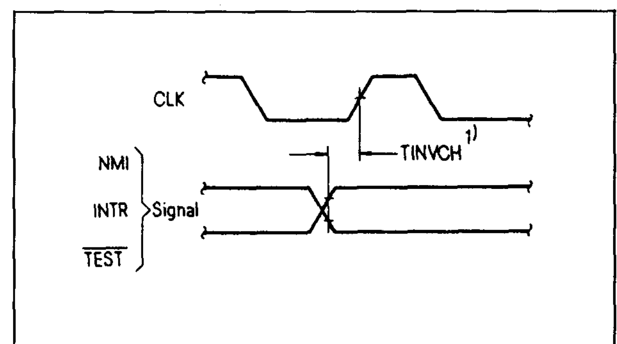
Figuur 7/3.1-11b: A.C. Test In- en uitgangssignalen.



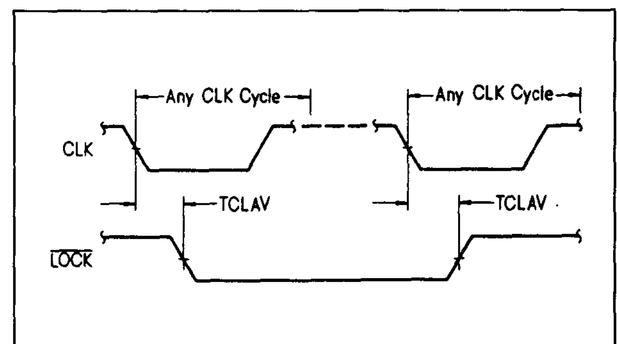
Figuur 7/3.1-11a: A.C. Testschakelingen.

Opslagtemperatuur: − 65 tot + 150 °C
Spanning op alle pennen ten opzichte van GND: − 1 tot + 7 V
Ingangsspanningen (80C86): − 2 tot (Vcc + 0,5) V
Uitgangsspanningen (80C86): − 0,5 tot (Vcc + 0,5) V
Dissipatie 8086: 2,5 W (80C86: 1 W)

Elektrische eigenschappen
In tabel 7/3.1-7 zijn de gelijkstroom-eigenschappen van de 80C86 te zien (die van de 8086 zijn bijna dezelfde, op het stroomverbruik en ingangscapaciteit na).
In tabel 7/3.1-8 en de bijbehorende figuren 7/3.1-9a en b zijn de minimum mode wisselspanningskarakteristieken voor de 80C86 weergegeven, terwijl tabel 7/3.1-9 en de figuren 7/3.1-10a en b hetzelfde voor de maximum mode laten zien. In de figuren 7/3.1-11a en b is te zien met welke schakelingen en golfvormen de processor werd getest.
De figuren 7/3.1-12 tot en met 15 laten de resterende belangrijke timingen zien.



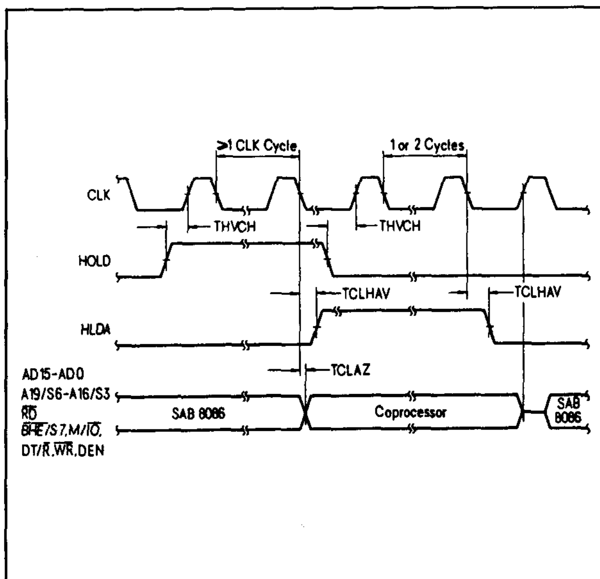
Figuur 7/3.1-12: Asynchrone signaalherkenning.



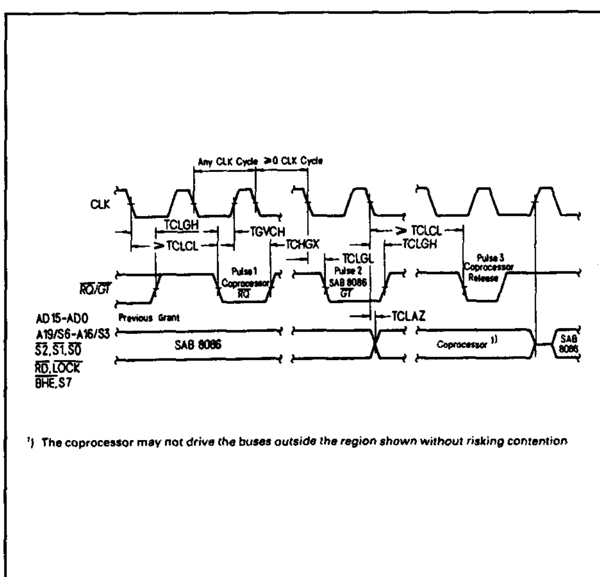
Figuur 7/3.1-13: Timing van het Bus Lock-signaal (alleen maximum mode).

De instructieset

Zie de samenvatting van de instructieset (tabel 7/3.1-10) en de toelichtingen hierop (tabellen 7/3.1-11, 12 en 13).



Figuur 7/3.1-14: Hold/Hold Acknowledge-timing (alleen minimum mode).



Figuur 7/3.1-15: Request/Grant-volgorde timing (maximum mode).

Key to Operand Types

IDENTIFIER	EXPLANATION
(no operands)	No operands are written
register	An 8- or 16-bit general register
reg 16	An 16-bit general register
seg-reg	A segment register
accumulator	Register AX or AL
immediate	A constant in the range 0-FFFFH
immed8	A constant in the range 0-FFH
memory	An 8- or 16-bit memory location[1]
mem8	An 8-bit memory location[1]
mem16	A 16-bit memory location[1]
source-table	Name of 256-byte translate table
source-string	Name of string addressed by register SI
dest-string	Name of string, addressed by register DI
DX	Register DX
short-label	A label within −128 to +127 bytes of the end of the instruction
near-label	A label in current code segment
far-label	A label in another code segment
near-proc	A procedure in current code segment
far-proc	A procedure in another code segment
memptr16	A word containing the offset of the location in the current code segment to which control is to be transferred[1]
memptr32	A doubleword containing the offset and the segment base address of the location in another code segment to which control is to be transferred[1]
regptr16	A 16-bit general register containing the offset of the location in the current code segment to which control is to be transferred
repeat	A string instruction repeat prefix

[1] **Any addressing mode — direct, register indirect, based, indexed, or based indexed — may be used**

Tabel 7/3.1-12: Toelichting op de instructieset (operand typen).

3.1 8086

Data Transfer

Instruction	76543210	76543210	76543210	76543210
MOV = Move:				
Register / memory to / from register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate to register	1 0 1 1 w reg	data	data if w=1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		
PUSH = Push:				
Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m		
Register	0 1 0 1 0 reg			
Segment register	0 0 0 reg 1 1 0			
POP = Pop:				
Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m		
Register	0 1 0 1 1 reg			
Segment register	0 0 0 reg 1 1 1			
XCHG = Exchange:				
Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m		
Register with accumulator	1 0 0 1 0 reg			
IN = Input from:				
Fixed port	1 1 1 0 0 1 0 w	port		
Variable port	1 1 1 0 1 1 0 w			

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor.

3.1 8086

	76543210	76543210	76543210	76543210
OUT = Output to:				
Fixed port	1 1 1 0 0 1 1 w	port		
Variable port	1 1 1 0 1 1 1 w			
XLAT = Translate byte to AL	1 1 0 1 0 1 1 1			
LEA = Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m		
LDS = Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		
LES = Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		
LAHF = Load AH with flags	1 0 0 1 1 1 1 1			
SAHF = Store AH into flags	1 0 0 1 1 1 1 0			
PUSHF = Push flags	1 0 0 1 1 1 0 0			
POPF = Pop flags	1 0 0 1 1 1 0 1			
Arithmetic				
ADD = Add:				
Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s: w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1	
ADC = Add with carry:				
Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1	
INC = Increment:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		
Register	0 1 0 0 0 reg			
AAA = ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA = Decimal adjust for add	0 0 1 0 0 1 1 1			

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor. *(Vervolg)*

3.1 8086

	76543210	76543210	76543210	76543210
SUB = Subtract:				
Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	
SBB = Subtract with borrow:				
Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	

	76543210	76543210	76543210	76543210
DEC = Decrement:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG = Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		
CMP = Compare:				
Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1	
AAS = ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS = Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL = Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL = Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV = Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV = Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW = Convert byte to word	1 0 0 1 1 0 0 0			
CWD = Convert word to double word	1 0 0 1 1 0 0 1			

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor. *(Vervolg)*

3.1 8086

	76543210	76543210	76543210	76543210
Logic				
NOT = Invert	1111011w	mod 010 r/m		
SHL/SAL = Shift logical/arithmetic left	110100vw	mod 100 r/m		
SHR = Shift logical right	110100vw	mod 101 r/m		
SAR = Shift arithmetic right	110100vw	mod 111 r/m		
ROL = Rotate left	110100vw	mod 000 r/m		
ROR = Rotate right	110100vw	mod 001 r/m		
RCL = Rotate through carry flag left	110100vw	mod 010 r/m		
RCR = Rotate through carry flag right	110100vw	mod 011 r/m		
AND = And:				
Reg./memory and register to either	001000dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 100 r/m	data	data if w=1
Immediate to accumulator	0010010w	data	data if w=1	
TEST = And function to flags, no result:				
Register/memory and register	1000010w	mod reg r/m		
Immediate data and register/memory	1111011w	mod 000 r/m	data	data if w=1
Immediate data and accumulator	1010100w	data	data if w=1	
OR = Or:				
Reg./memory and register to either	000010dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 001 r/m	data	data if w=1
Immediate to accumulator	0000110w	data	data if w=1	
XOR = Exclusive Or:				
Reg./memory and register to either	001100dw	mod reg r/m		
Immediate to register/memory	1000000w	mod 110 r/m	data	data if w=1
Immediate to accumulator	0011010w	data	data if w=1	

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor. *(Vervolg)*

3.1 8086

String Manipulation	76543210	76543210	76543210
REP = Repeat	1111001z		
MOVS = Move byte/word	1010010w		
CMPS = Compare byte/word	1010011w		
SCAS = Scan byte/word	1010111w		
LODS = Load byte/word to AL/AX	1010110w		
STOS = Store byte/word from AL/A	1010101w		
Control Transfer			
CALL = Call:			
Direct within segment	11101000	disp-low	disp-high
Indirect within segment	11111111	mod 010 r/m	
Direct intersegment	10011010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 011 r/m	
JMP = Unconditional jump:			
Direct within segment	11101001	disp-low	disp-high
Direct within segment short	11101011	disp	
Indirect within segment	11111111	mod 100 r/m	
Direct intersegment	11101010	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	11111111	mod 101 r/m	

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor. *(Vervolg)*

RET = Return from CALL:	**76543210**	**76543210**	**76543210**
Within segment	11000011		
Within seg. adding immediate to SP	11000010	data-low	data-high
Intersegment	11001011		
Intersegment adding immediate to SP	11001010	data-low	data-high
JE/JZ = Jump on equal/zero	01110100	disp	
JL/JNGE = Jump on less/not greater or equal	01111100	disp	
JLE/JNG = Jump on less or equal/not greater	01111110	disp	
JB/JNAE = Jump on below/not above or equal	01110010	disp	
JBE/JNA = Jump on below or equal/ not above	01110110	disp	
JP/JPE = Jump on parity/parity even	01111010	disp	
JO = Jump on overflow	01110000	disp	
JS = Jump on sign	01111000	disp	
JNE/JNZ = Jump on not equal/not zero	01110101	disp	
JNL/JGE = Jump on not less/greater or equal	01111101	disp	
JNLE/JG = Jump on not less or equal/ greater	01111111	disp	
JNB/JAE = Jump on not below/above or equal	01110011	disp	
JNBE/JA = Jump on not below or equal/above	01110111	disp	
JNP/JPO = Jump on not parity/parity odd	01111011	disp	
JNO = Jump on not overflow	01110001	disp	
JNS = Jump on not sign	01111001	disp	
LOOP = Loop CX times	11100010	disp	
LOOPZ/LOOPE = Loop while zero/equal	11100001	disp	
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000	disp	
JCXZ = Jump on CX zero	11100011	disp	

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor. *(Vervolg)*

3.1 8086

INT = Interrupt	**76543210**	**76543210**
Type specified	11001101	type
Type 3	11001100	
INTO = Interrupt on overflow	11001110	
IRET = Interrupt return	11001111	
Processor Control		
CLC = Clear carry	11111000	
CMC = Complement carry	11110101	
STC = Set carry	11111001	
CLD = Clear direction	11111100	
STD = Set direction	11111101	
CLI = Clear interrupt	11111010	
STI = Set interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to external device)	11011xxx	mod xxx r/m
LOCK = Bus lock prefix	11110000	

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor. *(Vervolg)*

3.1 8086

Footnotes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if s:w = 01 then 16-bits of immediate data from the operand
it s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparsion with ZF FLAG

Segment Override Prefix

0 0 1 reg 1 1 0

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp high is absent
if mod = 10 then DISP = disp-high: disp low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) +DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

* except if mod = 00 and r/m = 110 then EA = disp-high:disp-low.

REG is assigned according to the following table

16-bit (w=1)	8-bit (w=0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instruction which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

Tabel 7/3.1-10: Samenvatting van de instructieset van de 8086 processor. *(Vervolg)*

3.1 8086

"reg" Field Bit Assignments:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

"mod" Field Bit Assignments:

mod xxx r/m

mod	Displacement
00	DISP = 0*, disp-low and disp-high are absent
01	DISP = disp-low sign-extended to 16-bits, disp-high is absent
10	DISP = disp-high: disp-low
11	r/m is treated as a "reg" field

"r/m" Field Bit Assignments:

r/m	Operand Address
000	(BX) + (SI) + DISP
001	(BX) + (DI) + DISP
010	(BP) + (SI) + DISP
011	(BP) + (DI) + DISP
100	(SI) + DISP
101	(DI) + DISP
110	(BP) + DISP
111	(BX) + DISP

DISP follows 2nd byte of instruction (before data if required).
*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

Tabel 7/3.1-11: Toelichting op de instructieset van de 8086 (fields).

3.1 8086

IDENTIFIER	USED IN	EXPLANATION
destination	data transfer, bit manipulation	A register or memory location that may contain data operated on by the instruction, and which receives (is replaced by) the result of the operation.
source	data transfer, arithmetic, bit manipulation	A register, memory location or immediate value that is used in the operation, but is not altered by the instruction.
source-table	XLAT	Name of memory translation table addressed by register BX.
target	JMP, CALL	A label to which control is to be transferred directly, or a register or memory location whose *content* is the address of the location to which control is to be transferred indirectly.
short-label	cond. transfer, iteration control	A label to which control is to be conditionally transferred; must lie within −128 to +127 bytes of the first byte of the next instruction.
accumulator	IN, OUT	Register AX for word transfers, AL for bytes.
port	IN, OUT	An I/O port number; specified as an immediate value of 0-255, or register DX (which contains port number in range 0-64k).
source-string	string ops.	Name of a string in memory that is addressed by register SI; used only to identify string as byte or word and specify segment override, if any. This string is used in the operation, but is not altered.
dest-string	string ops.	Name of string in memory that is addressed by register DI; used only to identify string as byte or word. This string receives (is replaced by) the result of the operation.
count	shifts, rotates	Specifies number of bits to shift or rotate; written as immediate value 1 or register CL (which contains the count in the range 0-255).
interrupt-type	INT	Immediate value of 0-255 identifying interrupt pointer number.
optional-pop-value	RET	Number of bytes (0-64k, ordinarily an even number) to discard from stack.
external-opcode	ESC	Immediate value (0-63) that is encoded in the instruction for use by an external processor.
above-below	conditional jumps	Above and below refer to the relationship of two unsigned values.
greater-less	conditional jumps	Greater and less refer to the relationship of two signed values.

Tabel 7/3.1-13: Toelichting op de instructieset (idenrifiers).

7/3.3

80286

Inleiding

Algemeen

De 80286 is de microprocessor waarop de zogenaamde AT-typen Personal Computers zijn gebaseerd. De "desktop" AT's (bureaumodellen met grote monitor) zijn ondertussen wel over hun hoogtepunt heen en worden steeds vaker vervangen door "386"- en "486"-typen.
De CMOS-uitvoering, de 80C286 vindt voorlopig nog ruime toepassing in draagbare "laptop" (schoot)-computers. In het onderstaande wordt de 80286 microprocessor volledig behandeld.

De 80286 is een geavanceerde microprocessor met speciale, geoptimaliseerde functies voor toepassing door meerdere gebruikers tegelijk en voor multi-tasking systemen. De 80286 heeft een ingebouwde beveiliging van het geheugen ter ondersteuning van het bedrijfssysteem en de isolatie van taken, terwijl ook programma's en data binnen eenzelfde taak apart worden gehouden.

Een op 12 MHz werkende 80286 kan maximaal tienmaal zoveel presteren als een standaard 5 MHz 8086.
Met de 80286 kan maximaal 2^{30} bytes (één gigabyte) virtueel geheugen per taak worden geadresseerd in 2^{24} bytes (16 megabyte) fysiek geheugen.

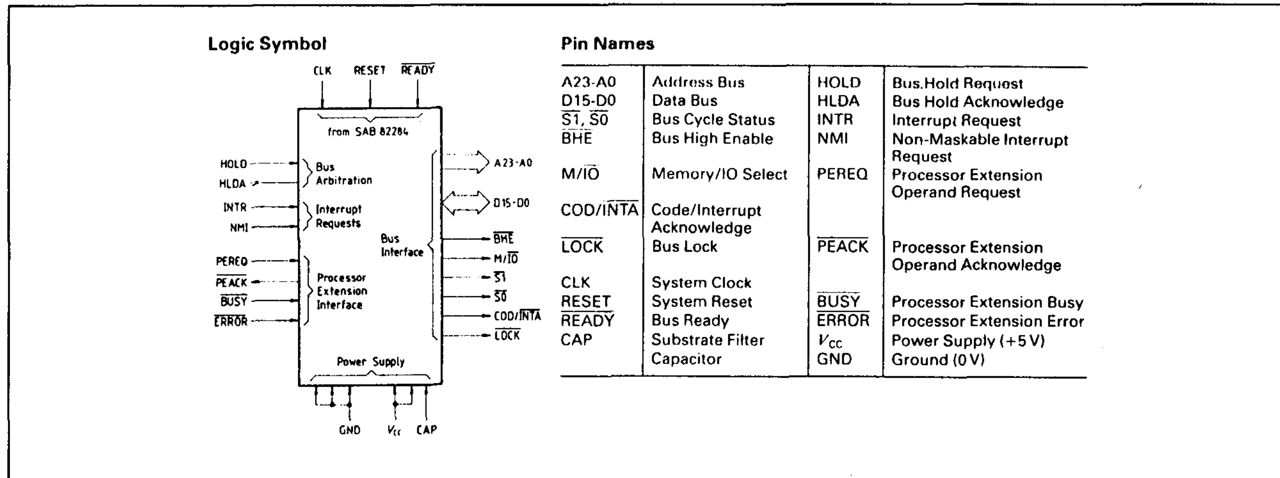
De 80286 is "opwaarts compatibel" voor de software van de 8086 en de 8088. Wanneer de reële adresseermode van de 8086 wordt gebruikt, is de 80286 object-code compatibel met software voor de 8088/8086. In de beveiligde virtuele adresseermode is de 80286 source code (broncode) compatibel met 8088/8086 software en kan alleen van de uitgebreide virtuele adressen gebruik worden gemaakt als de software wordt bijgewerkt.
In beide modes komt het prestatievermogen van de 80286 volledig tot zijn recht en wordt een superset van de 8086/8088 instructies uitgevoerd.

De 80286 kan speciale handelingen verrichten voor de ondersteuning van efficiënte implementatie en uitvoering van bedrijfssystemen.
Eén instructie is bijvoorbeeld voldoende om een lopende taak te beëindigen, de status ervan op te bergen, over te schakelen naar een nieuwe taak, de status daarvan te laden en met de uitvoering van deze nieuwe taak te beginnen.
De 80286 ondersteunt ook virtuele geheugensystemen met behulp van een segment-not-present uitzondering en herstartbare instructies.

Algemene gegevens

- fabrikanten en versies, o.a.:
  Intel: 80286 (-6, -8, -10, -12)
  Siemens: SAB 80286 (-1, -12)
  AMD: 80286 (-8, -10, -12, -16), 80L286
  Fujitsu: 80286 (-6, -8, -10)
  Harris: 80C286 (-10, -12, -16)
- grote adresseer-ruimte: 16 Mbyte fysiek, 1 Gbyte virtueel

3.3 80286



Pin Names

A23-A0	Address Bus	HOLD	Bus.Hold Request
D15-D0	Data Bus	HLDA	Bus Hold Acknowledge
$\overline{\text{S1}}$, $\overline{\text{S0}}$	Bus Cycle Status	INTR	Interrupt Request
$\overline{\text{BHE}}$	Bus High Enable	NMI	Non-Maskable Interrupt Request
M/$\overline{\text{IO}}$	Memory/IO Select	PEREQ	Processor Extension Operand Request
COD/$\overline{\text{INTA}}$	Code/Interrupt Acknowledge		
$\overline{\text{LOCK}}$	Bus Lock	$\overline{\text{PEACK}}$	Processor Extension Operand Acknowledge
CLK	System Clock		
RESET	System Reset	$\overline{\text{BUSY}}$	Processor Extension Busy
$\overline{\text{READY}}$	Bus Ready	$\overline{\text{ERROR}}$	Processor Extension Error
CAP	Substrate Filter Capacitor	V_{CC}	Power Supply (+5 V)
		GND	Ground (0 V)

Figuur 7/3.3-1: Logisch symbool en benamingen van de aansluitpennen.

- geïntegreerd geheugen-management, geheugenbescherming op vier niveaus en ondersteuning van virtueel geheugen en bedrijfssystemen
- bus-interface met grote bandbreedte: 12,5 Mbyte/s
- ondersteuning door industriële bedrijfssystemen, zoals iRMX, XENIX, UNIX, MS-DOS
- optionele uitbreiding met 80287 (80-bit numerieke coprocessor)
- 8086 opwaarts compatibele bedrijfsmodes: 8086 Real Address Mode en Protected Virtual Address Mode
- kloksnelheden: 6 MHz (-6 type) tot 16 MHz (-16 type)
- complete ondersteuning voor het ontwikkelen van systemen:
  - ontwikkel-software: assembler, PL/M, Pascal, Fortran en systeem-utilities;
  - in-circuit emulator: ICE-286
- behuizingen: 68-pens ceramisch LCC (Leadless Chip Carrier), PGA (Pin Grid Array) en PLCC (Plastic Leaded Chip Carrier)
- ondersteunende chips:
  80287: numerieke coprocessor
  8259A: interrupt controller
  82258: DMA controller
  82284: clock generator
  82288: bus-controller
  82289: bus-arbiter

Aansluitingen en pen-functies

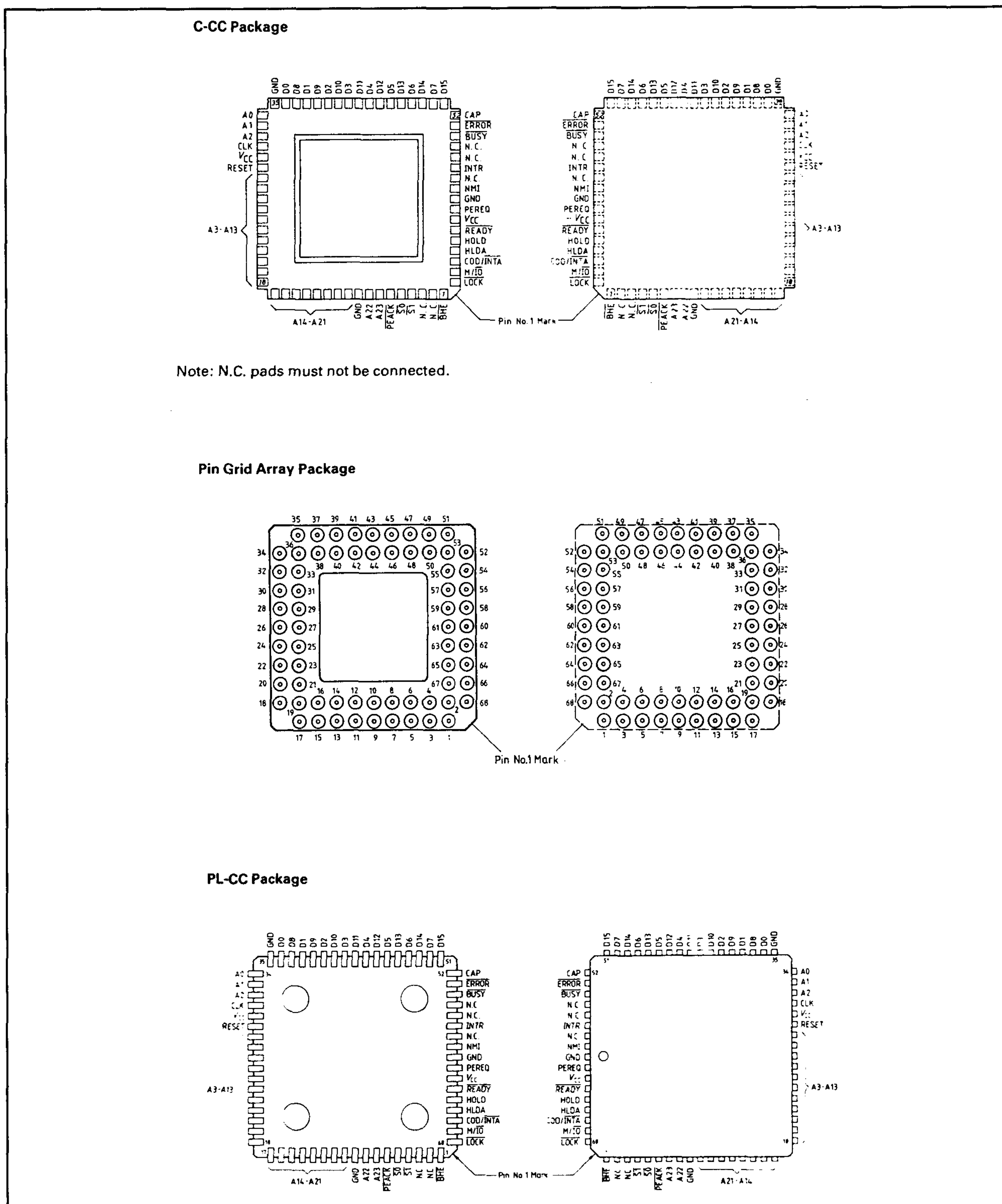
Inleiding

De 80286 is leverbaar in drie soorten behuizingen: een 68-pens C-CC, een PL-CC (ceramisch- of plastic leaded chip-carrier) of een 68-pens Pin Grid Array (PGA).
In figuur 7/3.3-2 zijn deze behuizingen getekend: aan de linkerzijde het onderaanzicht (op de pennen gezien) en aan de rechterzijde het bovenaanzicht (de componentenzijde van de print). De nummering en de benamingen van de aansluitingen (tabel 7/3.3-1) gelden voor alle drie de behuizingen.

$\overline{\text{BHE}}$, uitgang, pen 1

De BUS HIGH ENABLE-uitgang geeft aan dat data-transport op de hoogste byte (D15 tot en met D8) van de databus plaats vindt. Door 8-bit schakelingen die op de hoogste byte van de databus zijn aangesloten zal $\overline{\text{BHE}}$ meestal worden gebruikt voor chip-select functies.
$\overline{\text{BHE}}$ is aktief-LAAG (en hoog-impedant tijdens bus hold acknowledge).

3.3 80286



Figuur 7/3.3-2: Aansluitgegevens van de C-CC, PGA en PL-CC uitvoeringen van de 80286 microprocessor (links: onderzijde, rechts: bovenzijde).

3.3 80286

NAME	PAD	PIN	NAME	PAD	PIN
$\overline{BHE}$	1	B1	V_{SS}	35	K11
NC	2	B2	D_0	36	K10
NC	3	C1	D_8	37	J11
$\overline{S1}$	4	C2	D_1	38	J10
$\overline{S0}$	5	D1	D_9	39	H11
$\overline{PEACK}$	6	D2	D_2	40	H10
A_{23}	7	E1	D_{10}	41	G11
A_{22}	8	E2	D_3	42	G10
V_{SS}	9	F1	D_{11}	43	F11
A_{21}	10	F2	D_4	44	F10
A_{20}	11	G1	D_{12}	45	E11
A_{19}	12	G2	D_5	46	E10
A_{18}	13	H1	D_{13}	47	D11
A_{17}	14	H2	D_6	48	D10
A_{16}	15	J1	D_{14}	49	C11
A_{15}	16	J2	D_7	50	C10
A_{14}	17	K1	D_{15}	51	B11
A_{13}	18	L2	CAP	52	A10
A_{12}	19	K2	$\overline{ERROR}$	53	B10
A_{11}	20	L3	$\overline{BUSY}$	54	A9
A_{10}	21	K3	NC	55	B9
A_9	22	L4	NC	56	A8
A_8	23	K4	INTR	57	B8
A_7	24	L5	NC	58	A7
A_6	25	K5	NMI	59	B7
A_5	26	L6	V_{SS}	60	A6
A_4	27	K6	PEREQ	61	B6
A_3	28	L7	V_{CC}	62	A5
RESET	29	K7	$\overline{READY}$	63	B5
V_{CC}	30	L8	HOLD	64	A4
CLK	31	K8	HLDA	65	B4
A_2	32	L9	COD/$\overline{INTA}$	66	A3
A_1	33	K9	M/$\overline{IO}$	67	B3
A_0	34	L10	$\overline{LOCK}$	68	A2

Tabel 7/3.3-1: Pen-functies van de 80286.

$\overline{BHE}$ and A0 encodings		
$\overline{BHE}$ value	A0 value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus (D15-8)
1	0	Byte transfer on lower half of data bus (D7-0)
1	1	Reserved

Tabel 7/3.3-2: Coderingen van $\overline{BHE}$ en A0.

$\overline{S1}$, $\overline{S0}$, uitgangen, pennen 4 en 5

De BUS CYCLE STATUS-uitgangen geven het initialiseren van een buscyclus aan en bepalen samen met M/$\overline{IO}$ en COD/$\overline{INTA}$ het type van de buscyclus.

De bus is in de 3-state toestand (zwevend) als $\overline{S1}$ en/of $\overline{S2}$ LAAG is. $\overline{S1}$ en $\overline{S2}$ zijn aktief-LAAG en bevinden zich tijdens bus hold-acknowledge in de 3-state "uit" toestand.

Bus cycle status definition				
COD/$\overline{INTA}$	M/$\overline{IO}$	$\overline{S1}$	$\overline{S0}$	Bus cycle initiated
0 (low)	0	0	0	Interrupt acknowledge
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	None; not a status cycle
0	1	0	0	IF A1 = 1 then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (high)	0	0	0	Reserved
1	0	0	1	I/O read
1	0	1	0	I/O write
1	0	1	1	None; not a status cycle
1	1	0	0	Reserved
1	1	0	1	Memory instruction read
1	1	1	0	Reserved
1	1	1	1	None; not a status cycle

Tabel 7/3.3-3: Definities van de buscyclus-status.

A23 tot en met A0, uitgangen, pennen 7 tot en met 34

Met de uitgangssignalen op de ADDRESS BUS worden fysiek geheugen en I/O-poorten geadresseerd. A0 is LAAG wanneer data via de pennen D7 tot en met D0 moet worden overgebracht. A23 tot en met A10 zijn LAAG tijdens I/O-transfers.

De adresbus is aktief-HOOG en zwevend bij bus hold-acknowledge.

RESET, ingang, pen 29

De interne logika van de 80286 wordt met SYSTEM RESET leeg gemaakt. SYSTEM RESET is aktief-HOOG. De 80286 kan op elk willekeurig moment opnieuw worden geïnitialiseerd door een LAAG-naar-HOOG overgang op de RESET-ingang die langer dan 16 systeem-clockcycli aktief blijft. Wan-

neer RESET aktief is gaan de uitgangen van de 80286 naar de in tabel 7/3.3-4 aangegeven toestanden.

Pin state during reset	
Pin value	Pin names
1 (high)	$\overline{S0}$, $\overline{S1}$, $\overline{PEACK}$, A23–A0, $\overline{BHE}$, $\overline{LOCK}$
0 (low)	M/$\overline{IO}$, COD/$\overline{INTA}$, HLDA
Tristate off	D15 – D0

Tabel 7/3.3-4: Toestanden van de pennen tijdens het resetten.

De 80286 begint met werken na een HOOG-naar-LAAG overgang op RESET. Deze overgang moet synchroon met de systeemclock gaan. Na de dalende flank van RESET zijn ongeveer 50 systeem-clockcycli nodig voor interne initialisatie voordat de eerste buscyclus wordt uitgevoerd (ophalen van code uit het power-on executie-adres).
Door een synchroon met de systeemclock optredende LAAG-naar-HOOG overgang van RESET wordt een processorcyclus beëindigd op de tweede HOOG-naar-LAAG overgang van de systeemclock. De LAAG-naar-HOOG overgang van RESET mag asynchroon met de systeemclock optreden, maar in dat geval kan niet worden voorspeld welke fase van de processorclock zal optreden gedurende de volgende periode van de systeemclock.
Synchrone LAAG-naar-HOOG overgangen zijn alleen vereist in systemen waarbij de processorclock fase-synchroon moet zijn met een andere clock.

CLK, ingang, pen 31
SYSTEM CLOCK levert de fundamentele timing van 80286 systemen. Dit ingangssignaal wordt in de 80286 door twee gedeeld voor het opwekken van de processorclock. De inwendige halveringsschakeling kan door een LAAG-naar-HOOG overgang op de RESET-ingang op een externe clockgenerator worden gesynchroniseerd.

D15 tot en met D0, in-/uitgangen, pen 36 tot en met 51
Tijdens leescycli voor geheugen-, I/O- en interrupt-acknowledge wordt data via de DATABUS D15 tot en met D0 naar de processor gebracht, terwijl data wordt afgegeven tijdens schrijfcycli voor geheugen en I/O. De databus is aktief-HOOG en bevindt zich in de zwevende 3-state "uit" toestand gedurende bus hold-acknowledge.

$\overline{ERROR}$ en $\overline{BUSY}$, ingangen, pen 53 en 54
De PROCESSOR EXTENSION BUSY- en ERROR-ingangen geven de 80286 informatie over de bedrijfsconditie van een processor-uitbreiding. Een aktief $\overline{BUSY}$-signaal laat de 80286 op WAIT en sommige ESC-instructies stoppen met de uitvoering van een programma totdat $\overline{BUSY}$ niet-aktief (= HOOG) wordt. De 80286 mag tijdens het wachten op het niet-aktief worden van $\overline{BUSY}$ worden geïnterrumpeerd.
Een aktief $\overline{ERROR}$-ingangssignaal heeft tot gevolg dat de 80286 een processor-uitbreidingsinterrupt geeft wanneer WAIT of bepaalde ESC-instructies worden uitgevoerd. Deze ingangen zijn aktief-LAAG en mogen asynchroon ten opzichte van de systeemclock werken.

INTR, ingang, pen 57
Met het INTERRUPT REQUEST-signaal wordt de 80286 verzocht om de uitvoering van het lopende programma uit te stellen om eerst service te verlenen aan een extern verzoek. Interrupt-requests worden gemaskeerd door het interrupt-enable bit in het flag-woord te clearen. Wanneer de 80286 op een interrupt-request reageert, voert hij twee interrupt-acknowledge buscycli uit om een 8-bit interrupt-vector in te lezen die de herkomst van de interruptie aangeeft. Om te garanderen dat het programma wordt onderbroken moet INTR aktief blijven totdat de eerste interrupt-acknowledge cyclus is volbracht. INTR wordt aan het begin van elke processorcyclus afgetast en moet minstens

3.3 80286

twee processorcycli vóór beëindiging van de lopende instructie aktief-HOOG zijn om vóór de volgende instructie te kunnen interrumperen. INTR is niveau-gevoelig, aktief-HOOG en mag asynchroon met de systeemclock optreden.

NMI, ingang, pen 59
Het NON-MASKABLE INTERRUPT REQUEST-signaal interrumpeert de 80286 met een intern geleverde vectorwaarde van 2. Er worden nu geen interrupt-acknowledge cycli uitgevoerd. De interrupt-enable bit in het flagwoord van de 80286 heeft geen invloed op deze ingang. Het NMI ingangssignaal is aktief-HOOG, mag asynchroon zijn ten opzichte van de systeemclock en wordt flankgetriggerd na interne synchronisatie. Voor een goede herkenning dient het signaal eerst tenminste vier cycli van de systeemclock LAAG te zijn geweest, waarna het minstens vier systeem clockcycli HOOG moet blijven.

PEREQ, ingang en
PEACK, uitgang, pen 61 en 6
De PROCESSOR EXTENSION OPERAND REQUEST-ingang en de PROCESSOR EXTENSION ACKNOWLEDGE-uitgang vergroten de mogelijkheden van geheugenmanagement en beveiliging van de 80286 naar processor-uitbreidingen. Het PEREQ-signaal vraagt de 80286 een data operand-transfer uit te voeren voor een processor-uitbreiding. De $\overline{\text{PEACK}}$-uitgang signaleert aan de processor-uitbreiding wanneer de gevraagde operand wordt overgebracht. PEREQ is aktief-HOOG. $\overline{\text{PEACK}}$ is aktief-LAAG en bevindt zich gedurende bus hold-acknowledge in de zwevende 3-state "uit" toestand. $\overline{\text{PEACK}}$ mag asynchroon zijn ten opzichte van de systeemclock.

$\overline{\text{READY}}$, ingang, pen 63
Met het BUS READY-signaal wordt een buscyclus beëindigd. Buscycli kunnen onbegrensd worden verlengd totdat zij worden beëindigd door het LAAG gaan van $\overline{\text{READY}}$. $\overline{\text{READY}}$ is een aktief-LAGE synchrone ingang die aan de systeemclock gerefereerde setup- en houdtijden nodig heeft om correct te werken.
Gedurende bus hold-acknowledge wordt $\overline{\text{READY}}$ genegeerd.

HOLD, ingang en
HLDA, uitgang, pen 65 en 64
BUS HOLD REQUEST en HOLD ACKNOWLEDGE regelen het bezit van de lokale 80286-bus. Met een HOLD-signaal kan een andere lokale busmaster om besturing van de lokale bus vragen. Wanneer dit is toegestaan zet de 80286 zijn busdrivers in de 3-state "uit"-toestand en aktiveert dan HLDA, waardoor de bus hold-acknowledge conditie wordt bereikt. De lokale bus blijft toegewezen aan de vragende master totdat HOLD niet-aktief wordt. Hierdoor wordt HLDA gedeactiveerd en neemt de 80286 de besturing van de lokale bus weer over. HOLD mag asynchroon ten opzichte van de systeemclock zijn. Beide signalen zijn aktief-HOOG.

COD/$\overline{\text{INTA}}$, uitgang, pen 66
Met CODE/INTERRUPT ACKNOWLEDGE wordt onderscheid gemaakt tussen instructie ophaal-cycli en geheugen data-leescycli. Tevens worden interrupt-acknowledge cycli onderscheiden van I/O-cycli.
De COD/$\overline{\text{INTA}}$-uitgang is 3-state "uit" (zwevend) gedurende bus hold-acknowledge.

M/$\overline{\text{IO}}$, uitgang, pen 67
MEMORY / I/O SELECT geeft het verschil aan tussen geheugen-toegang en I/O-toegang. Indien deze uitgang tijdens T_S HOOG is, is er een geheugen-cyclus of een halt/shutdown-cyclus bezig. Is de uitgang LAAG, dan loopt een I/O-cyclus of een interrupt-acknowledge cyclus. M/$\overline{\text{IO}}$ is 3-state "uit" gedurende bus-hold acknowledge.

$\overline{\text{LOCK}}$, uitgang, pen 68
BUS LOCK geeft aan dat het andere systeem-busmasters verboden is na de lopende buscyclus controle over de systeembus te krijgen. Het $\overline{\text{LOCK}}$-signaal kan expliciet wor-

3.3 80286

den geaktiveerd door de "LOCK" instructie-prefix of automatisch door 80286 hardware tijdens geheugen XCHG-instructies, interrupt-acknowledge of toegang tot de sleutelwoordentabel.
LOCK is aktief-LAAG en 3-state "uit" (zwevend) gedurende bus- hold acknowledge.

V_{CC}, pennen 30 en 62
De POWER SUPPLY-ingang wordt aangesloten op de +5 V systeemvoeding (+/-5 %).

V_{SS} (GND), pennen 9, 35 en 60
De GROUND-ingang wordt verbonden met de systeem-massa (aarde: 0 V).

CAP, ingang, pen 52
Een SUBSTRATE FILTER CONDENSATOR van 0,047 μF (+/-20 %, 12 V) moet tussen deze pen en aarde worden aangesloten. Deze condensator filtert het uitgangssignaal van de inwendige substraat bias-generator. De condensator mag een maximale DC lekstroom van 1 μA hebben.
Om correct te kunnen werken moet de substraat bias-generator van de 80286 deze condensator tot de werkspanning opladen. De oplaadtijd van de condensator bedraagt 5 ms (max.) nadat V_{CC} en CLK hun gespecificeerde DC en AC parameters hebben bereikt.
Gedurende deze tijd mag een RESET worden gegeven om verkeerd werken van de CPU te voorkomen. Na deze tijd kan de fase van de processorclock van de 80286 worden gesynchroniseerd op een andere clock door RESET synchroon met de systeemclock LAAG te pulseren.

Functionele beschrijving

Interne architectuur

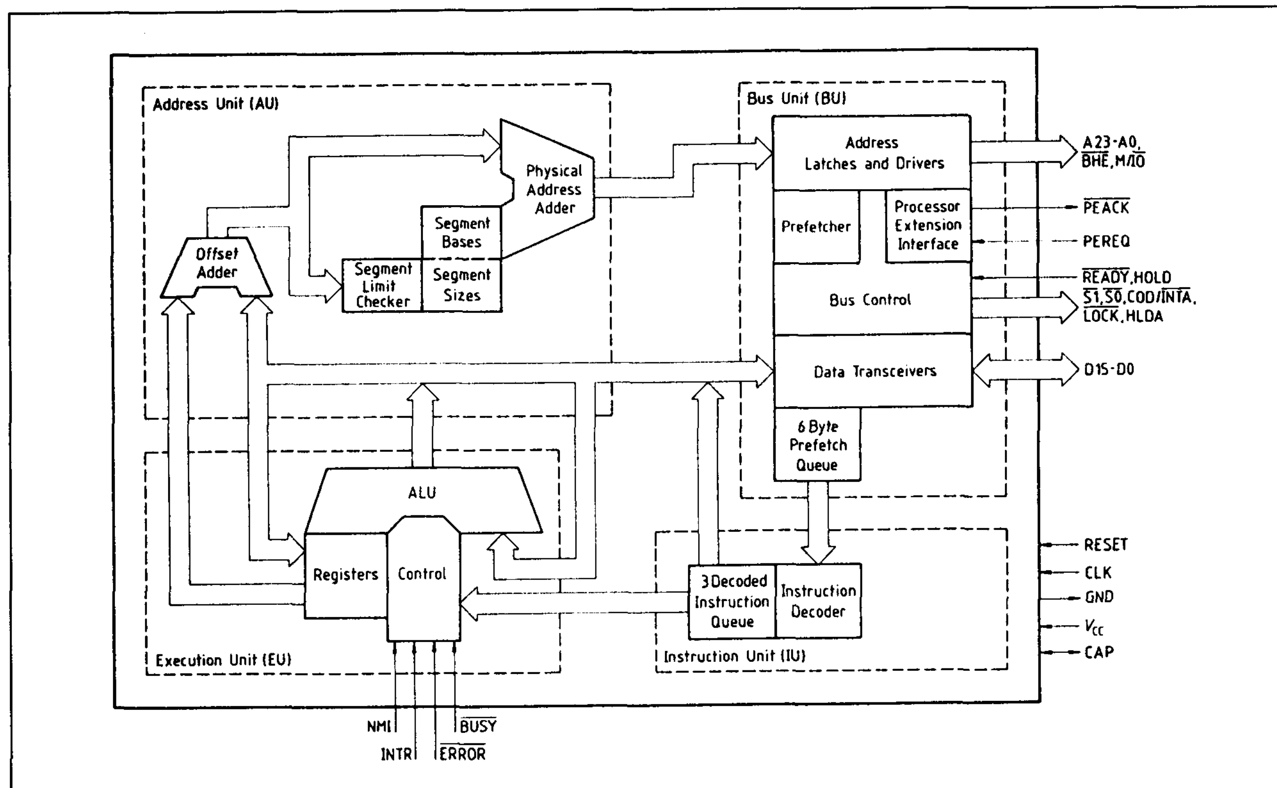
In figuur 7/3.3-3, het blokschema van de 80286, wordt de inwendige opbouw van de processor getoond. Er kunnen vier belangrijke delen worden onderscheiden:
- de Address Unit (AU);
- de Execution Unit (EU);
- de Bus Unit (BU);
- de Instruction Unit (IU).

Alle processoren van de 80xx-familie bevatten dezelfde set registers en maken gebruik van dezelfde instructies en adresseermodes. De 80286 processor is daardoor opwaarts compatibel met de CPU's 8088, 8086 en 80186.

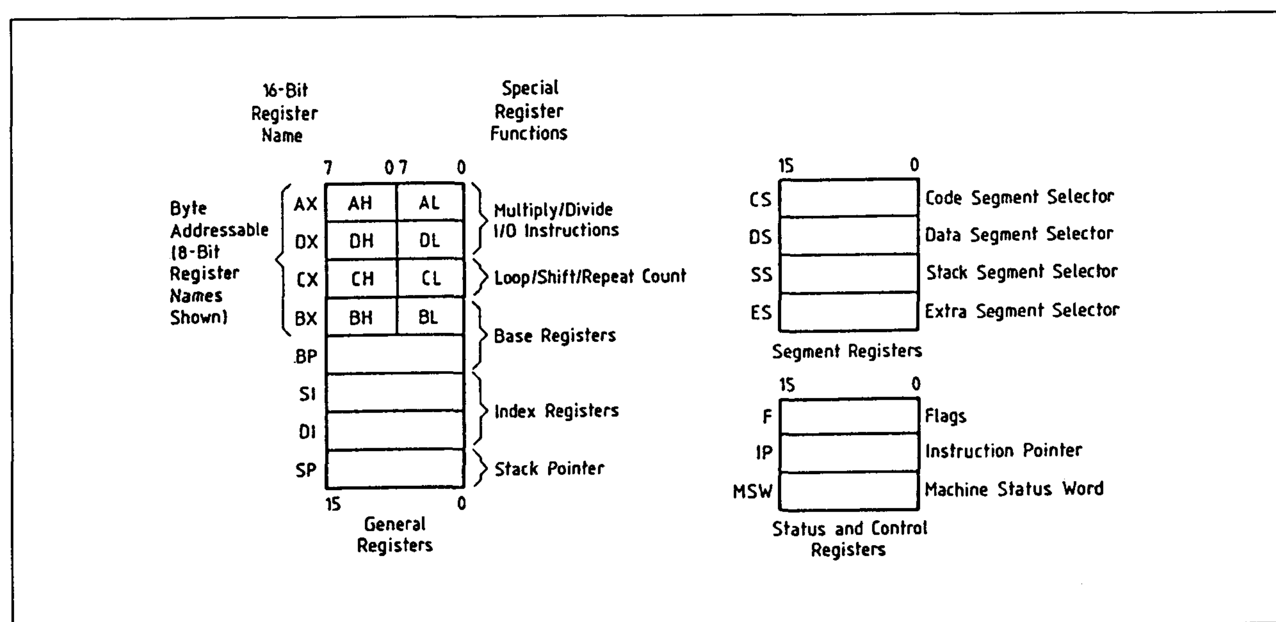
Registerset

De basisarchitectuur van de 80286 bevat vijftien registers (zie figuur 7/3.3-4) die kunnen worden onderverdeeld in de volgende categorieën:
- General registers
  Er zijn acht 16-bits algemeen bruikbare registers voor de opslag van rekenkundige en logische operands. Vier ervan (AX, BX, CX en DX) kunnen in hun geheel als 16-bits woorden worden gebruikt of opgesplitst in paren 8-bits registers.
- Segment registers
  De 80286 bevat vier speciale 16-bits registers die op elk willekeurig moment de geheugen-segmenten selecteren die onmiddellijk adresseerbaar zijn voor code, stack en data. Voor gebruik: zie Geheugen-organisatie.
- Basis- en index registers
  Vier van de algemeen toepasbare registers kunnen ook worden gebruikt voor het bepalen van offset-adressen van operands in het geheugen. Deze registers bevatten dan basis-adressen of indexen naar bijzondere lokaties binnen een segment.
  De adresseermode bepaalt welke specifieke registers voor het berekenen van operand-adressen worden gebruikt.
- Status- en control registers
- De drie speciale 16-bits registers in figuur 7/3.3-4 dienen voor het bijhouden of besturen van bepaalde aspecten van de 80286 processor-status, inclusief de instructie-pointer die het offset-adres van de volgende uit te voeren instructie bevat.

3.3 80286

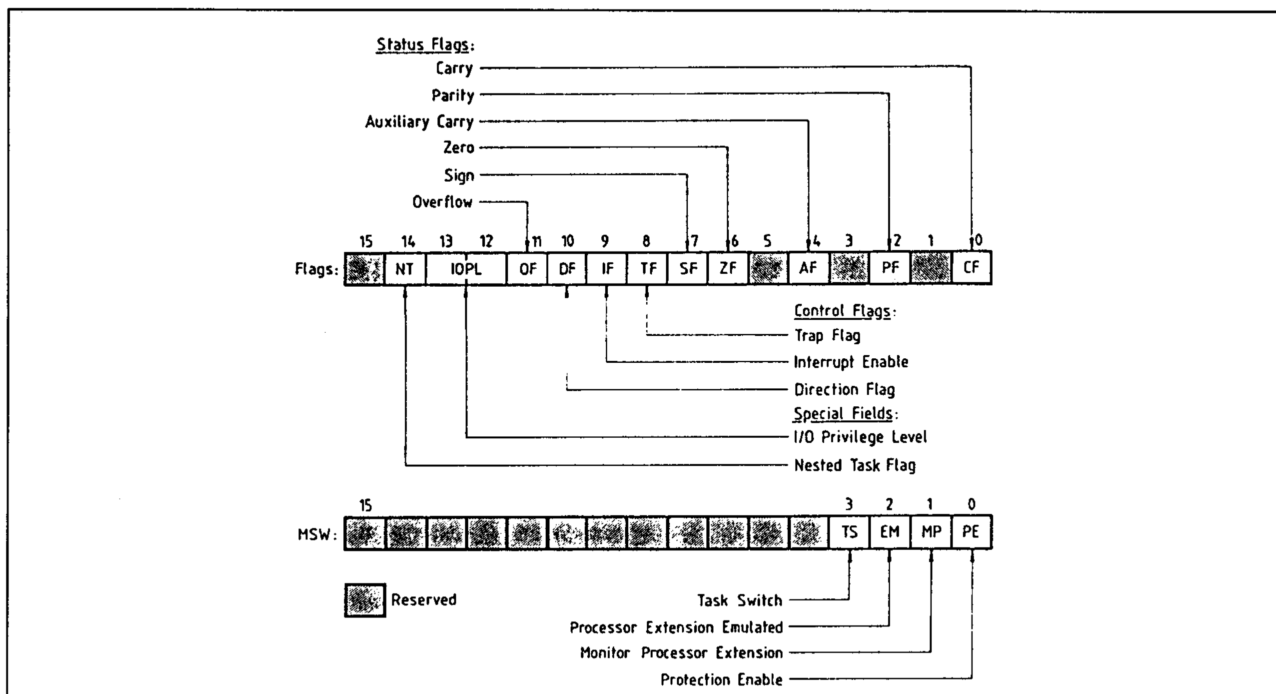


Figuur 7/3.3-3: Functioneel blokschema van de 80286.



Figuur 7/3.3-4: De registerset van een 80286 processor.

3.3 80286



Figuur 7/3.3-5: De positie van de besturingsbits in de status- en control-registers (boven: Flags-register, onder: het Machine Status Woord).

Beschrijving van het Flags-woord

In figuur 7/3.3-5 zijn het flag-register en het machine statuswoord apart genomen om hierin de plaats van de belangrijke status- en controlbits aan te geven.
De resultaten van logische en rekenkundige bewerkingen worden bijgehouden met de bits 0, 2, 4, 7 en 11 van het flag-register, terwijl de werking van de 80286 binnen een bepaalde bedrijfsmode afhankelijk is van de bits 8 en 9. In tabel 7/3.3-5 wordt de betekenis van de flag-bits verklaard.

Instructieset

De instructieset is opgedeeld in zeven categorieën:
- data-overdracht;
- rekenkundig;
- schuif/roteer/logisch;
- string manipulatie;
- besturings-overdracht;
- high-level instructies;
- processor besturing.

Deze categorieën zijn in tabel 7/3.3-6 (a tot en met g) samengevat.

Bit Position	Name	Function
0	CF	Carry Flag—Set on high-order bit carry or borrow; cleared otherwise
2	PF	Parity Flag—Set if low-order 8 bits of result contain an even number of 1-bits; cleared otherwise
4	AF	Set on carry from or borrow to the low order four bits of AL; cleared otherwise
6	ZF	Zero Flag—Set if result is zero; cleared otherwise
7	SF	Sign Flag—Set equal to high-order bit of result (0 if positive, 1 if negative)
11	OF	Overflow Flag—Set if result is a too-large positive number or a too-small negative number (excluding sign-bit) to fit in destination operand; cleared otherwise
8	TF	Single Step Flag—Once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt.
9	IF	Interrupt-enable Flag—When set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.
10	DF	Direction Flag—Causes string instructions to auto decrement the appropriate index registers when set. Clearing DF causes auto increment.

Tabel 7/3.3-5: Functies van de Flag-woord bits.

3.3 80286

GENERAL PURPOSE	
MOV	Move byte or word
PUSH	Push word onto stack
POP	Pop word off stack
PUSHA	Push all registers on stack
POPA	Pop all registers from stack
XCHG	Exchange byte or word
XLAT	Translate byte
INPUT/OUTPUT	
IN	Input byte or word
OUT	Output byte or word
ADDRESS OBJECT	
LEA	Load effective address
LDS	Load pointer using DS
LES	Load pointer using ES
FLAG TRANSFER	
LAHF	Load AH register from flags
SAHF	Store AH register in flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack

Tabel 7/3.3-6a: Data-overdracht instructies.

ADDITION	
ADD	Add byte or word
ADC	Add byte or word with carry
INC	Increment byte or word by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
SUBTRACTION	
SUB	Subtract byte or word
SBB	Subtract byte or word with borrow
DEC	Decrement byte or word by 1
NEG	Negate byte or word
CMP	Compare byte or word
AAS	ASCII adjust for subtraction
DAS	Decimal adjust for subtraction
MULTIPLICATION	
MUL	Multiple byte or word unsigned
IMUL	Integer multiply byte or word
AAM	ASCII adjust for multiply
DIVISION	
DIV	Divide byte or word unsigned
IDIV	Integer divide byte or word
AAD	ASCII adjust for division
CBW	Convert byte to word
CWD	Convert word to doubleword

Tabel 7/3.3-6b: Rekenkundige instructies.

Een 80286 instructie kan betrekking hebben op nul, een of twee operands. Een operand kan daarbij in een register, in de instructie zelf of in geheugen zijn opgeslagen.
Nul-operand instructies (bijvoorbeeld NOP en HLT) zijn meestal één byte lang. Eén-operand instructies (bijvoorbeeld INC en DEC) zijn vaak twee bytes lang, maar sommige worden in slechts één byte gecodeerd. Eén-operand instructies hebben betrekking op een register of een geheugen-lokatie.
Twee-operand instructies maken de volgende zes soorten handelingen mogelijk:
- register naar register;
- geheugen naar register;
- onmiddellijk (immediate) naar register;
- geheugen naar geheugen;
- register naar geheugen;
- onmiddellijk (immediate) naar geheugen.

Twee-operand instructies (bijvoorbeeld MOV en ADD) zijn gewoonlijk drie tot zes bytes lang. Geheugen naar geheugen operaties worden verzorgd door een speciale klasse string-instructies waarvoor slechts één tot drie bytes nodig zijn. Voor gedetailleerde instructieformaten en coderingen wordt verwezen naar de samenvatting van de instructieset aan het einde van dit hoofdstuk.

MOVS	Move byte or word string
INS	Input bytes or word string
OUTS	Output bytes or word string
CMPS	Compare byte or word string
SCAS	Scan byte or word string
LODS	Load byte or word string
STOS	Store byte or word string
REP	Repeat
REPE/REPZ	Repeat while equal/zero
REPNE/REPNZ	Repeat while not equal/not zero

Tabel 7/3.3-6c: String instructies.

Geheugen organisatie

Het geheugen is georganiseerd als sets segmenten met variabele lengte. Elk segment is een lineaire, aangrenzende reeks van maximaal 64 k (2^{16}) 8-bit bytes.
Het geheugen wordt geadresseerd door middel van een uit twee componenten bestaand adres (een pointer) die bestaat uit een 16-bit

3.3 80286

segment selector en een 16-bit offset. De segment selector geeft het gewenste geheugen-segment aan, terwijl de offset-component het gewenste byte-adres binnen het segment aanwijst (figuur 7/3.3-6).

Alle instructies die operands in het geheugen adresseren moeten het segment en de offset specificeren. Om een hoge snelheid en een compacte codering van de instructies te bereiken worden segment-selectoren meestal opgeslagen in de high-speed segment-registers. Een instructie hoeft slechts het verlangde segment-register en een offset te specificeren om een geheugen-operand te adresseren.
Voor de meeste instructies is het niet nodig expliciet te specificeren welk segment-register wordt gebruikt. Volgens de regels van tabel 7/3.3-7 wordt automatisch het juiste segment-register gekozen. Deze regels volgen de manier waarop programma's worden geschreven (zie figuur 7/3.3-7) als onafhankelijke modules die gebieden voor code en data, een stack en toegang tot externe data-gebieden nodig hebben.
Speciale segmenten hebben voorrang op instructie-prefixen, waardoor in speciale gevallen aan de impliciete selectieregels voor de segment-registers kan worden voorbij gegaan. De stack, data en extra segmenten mogen bij eenvoudige programma's gemeenschappelijk worden gebruikt.
Om toegang te krijgen tot operands die zich niet in een van de vier onmiddellijk verkrijgbare segmenten bevinden moet een complete 32-bit pointer of een nieuwe segment-selector worden geladen.

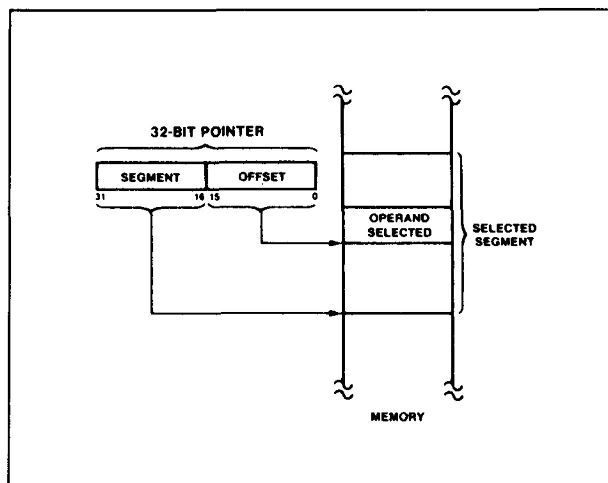
LOGICALS	
NOT	"Not" byte or word
AND	"And" byte or word
OR	"Inclusive or" byte or word
XOR	"Exclusive or" byte or word
TEST	"Test" byte or word
SHIFTS	
SHL/SAL	Shift logical/arithmetic left byte or word
SHR	Shift logical right byte or word
SAR	Shift arithmetic right byte or word
ROTATES	
ROL	Rotate left byte or word
ROR	Rotate right byte or word
RCL	Rotate through carry left byte or word
RCR	Rotate through carry right byte or word

Tabel 7/3.3-6d: Schuif/roteer/logische instructies.

CONDITIONAL TRANSFERS		UNCONDITIONAL TRANSFERS	
JA/JNBE	Jump if above/not below nor equal	CALL	Call procedure
JAE/JNB	Jump if above or equal/not below	RET	Return from procedure
JB/JNAE	Jump if below/not above nor equal	JMP	Jump
JBE/JNA	Jump if below or equal/not above		
JC	Jump if carry	**ITERATION CONTROLS**	
JE/JZ	Jump if equal/zero	LOOP	Loop
JG/JNLE	Jump if greater/not less nor equal		
JGE/JNL	Jump if greater or equal/not less	LOOPE/LOOPZ	Loop if equal/zero
JL/JNGE	Jump if less/not greater nor equal	LOOPNE/LOOPNZ	Loop if not equal/not zero
JLE/JNG	Jump if less or equal/not greater	JCXZ	Jump if register CX = 0
JNC	Jump if not carry		
JNE/JNZ	Jump if not equal/not zero	**INTERRUPTS**	
JNO	Jump if not overflow	INT	Interrupt
JNP/JPO	Jump if not parity/parity odd		
JNS	Jump if not sign	INTO	Interrupt if overflow
JO	Jump if overflow	IRET	Interrupt return
JP/JPE	Jump if parity/parity even		
JS	Jump if sign		

Tabel 7/3.3-6e: Besturings-overdracht instructies.

3.3 80286



Figuur 7/3.3-6: Een adres wordt samengesteld uit twee componenten.

FLAG OPERATIONS	
STC	Set carry flag
CLC	Clear carry flag
CMC	Complement carry flag
STD	Set direction flag
CLD	Clear direction flag
STI	Set interrupt enable flag
CLI	Clear interrupt enable flag
EXTERNAL SYNCHRONIZATION	
HLT	Halt until interrupt or reset
WAIT	Wait for $\overline{BUSY}$ not active
ESC	Escape to extension processor
LOCK	Lock bus during next instruction
NO OPERATION	
NOP	No operation
EXECUTION ENVIRONMENT CONTROL	
LMSW	Load machine status word
SMSW	Store machine status word

Tabel 7/3.3-6f: Processor besturings instructies.

Adresseer-modes

De 80286 kent in totaal acht adresseer-modes voor instructies om operands te specificeren.

Voor instructies die werken op registers of onmiddellijke operands staan twee adresseer-modes ter beschikking:

- Register Operand Mode
  De operand bevindt zich in één van de 8- of 16-bit algemene registers.
- Immediate Operand Mode
  De operand is opgenomen in de instructie.

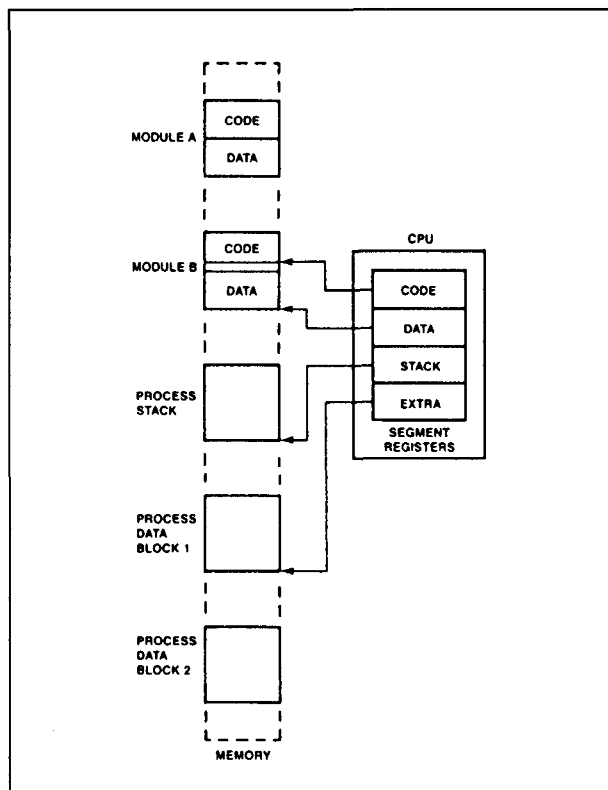
ENTER	Format stack for procedure entry
LEAVE	Restore stack for procedure exit
BOUND	Detects values outside prescribed range

Tabel 7/3.3-6g: High-level instructies.

Memory Reference Needed	Segment Register Used	Implicit Segment Selection Rule
Instructions	Code (CS)	Automatic with instruction prefetch
Stack	Stack (SS)	All stack pushes and pops. Any memory reference which uses BP as a base register.
Local Data	Data (DS)	All data references except when relative to stack or string destination
External (Global) Data	Extra (ES)	Alternate data segment and destination of string operation

Tabel 7/3.3-7: Selectieregels voor de segment-registers.

3.3 80286



Figuur 7/3.3-7: Door gesegmenteerd geheugen wordt de structuur van de software ondersteund.

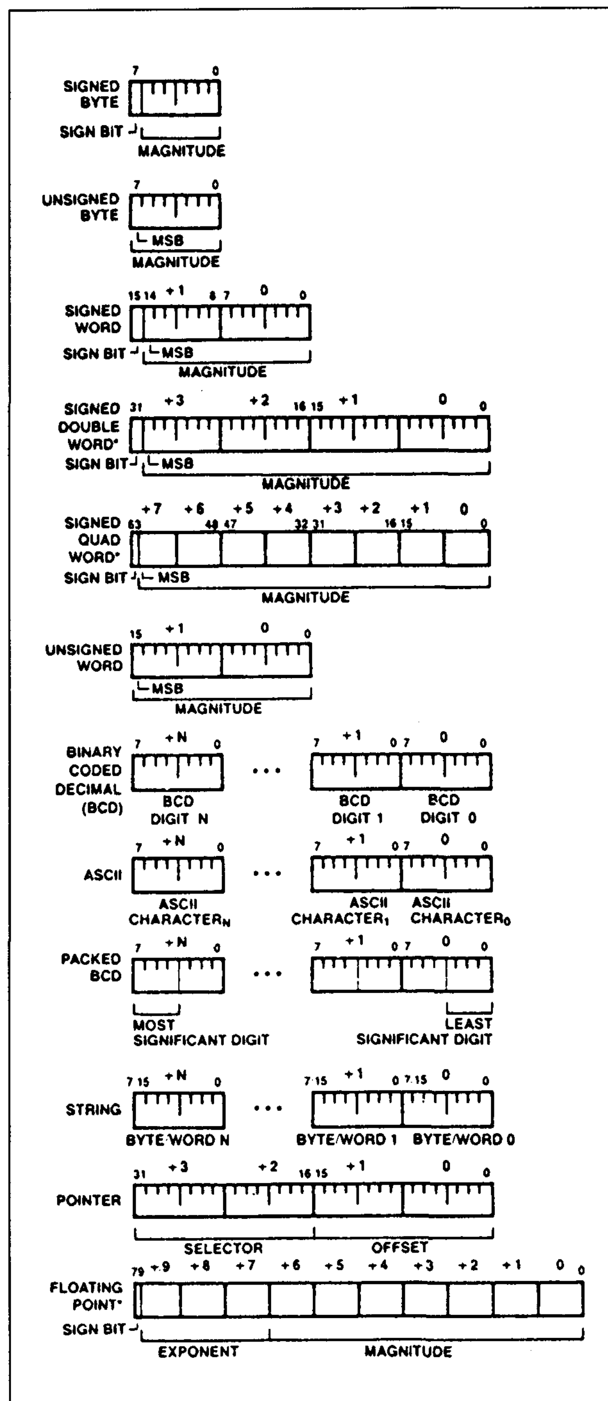
Om de plaats van een operand in een geheugen-segment te specificeren zijn zes modes voorhanden. Een geheugen operand-adres bestaat uit twee 16-bit delen: segment-selector en offset.
De segment-selector wordt geleverd door een segment-register dat óf impliciet is gekozen met de adresseer-mode óf expliciet met een "segment-override" prefix.
De offset wordt berekend door optellen van één van de volgende combinaties van adres-elementen:
- de ***verplaatsing*** (displacement: een 8- of 16-bit immediate waarde die in de instructie is opgenomen);
- de ***basis*** (base: inhoud van het BX of BP basis-register);
- de ***index*** (inhoud van het SI of DI index-register).

Elke carry uit de 16-bit optelling wordt gegeerd, terwijl 8-bit verplaatsingen worden uitgerekt tot 16-bit waarden (inclusief teken). Combinaties van deze drie adres-elementen bepalen de zes geheugen-adresseermodes die hieronder worden beschreven.
- Direct Mode
  de offset van de operand bevindt zich in de instructie als een 8- of 16-bit verplaatsingselement.
- Register Indirect Mode
  de offset van de operand zit in een van de registers SI, DI, BX of BP.
- Based Mode
  de offset van de operand is de som van een 8- of 16-bit verplaatsing en de inhoud van een basis-register (BX of BP).
- Indexed Mode
  de offset van de operand is de som van een 8- of 16-bit verplaatsing en de inhoud van een index-register (SI of DI).
- Based Indexed Mode
  de offset van de operand is de som van de inhoud van een basis-register en een index-register.
- Based Indexed Mode with Displacement
  de offset van de operand is de som van de inhoud van een basis-register, de inhoud van een index-register en een 8- of 16-bit verplaatsing.

Data Typen

De 80286 ondersteunt direct de volgende soorten data:
- Integer
  Een binaire numerieke waarde met teken die in een 8-bit byte of 16-bit woord staat. Alle handelingen gaan uit van een 2's complement representatie. 32- en 64-bit integers met teken worden ondersteund door de numerieke dataprocessor, de 80287.
- Ordinal (rangtelwoord):
  Een binaire numerieke waarde zonder teken, opgeslagen in een 8-bit byte of 16-bit woord.

3.3 80286



Figuur 7/3.3-8: De door de 80286 processor ondersteunde soorten data.

- Pointer
  Een 32-bit hoeveelheid, bestaande uit een segment-selector component en een offset component. Elke component is een 16-bit woord.
- String
  Een aansluitende reeks bytes of woorden. Een string kan 1-byte tot 64Kbytes bevatten.
- ASCII
  Een byte representatie van alfanumerieke en besturingskarakters, waarbij gebruik wordt gemaakt van de ASCII standaard karakter representatie.
- BCD
  Een (ongepakte) byte representatie van de decimale cijfers 0 tot en met 9.
- Packed BCD
  Een (gepakte) representatie van twee decimale cijfers 0 - 9, waarbij in elke nibble van de byte een cijfer wordt opgeslagen.
- Floating Point
  Representatie van een 32-, 64- of 80-bit reëel getal met teken. Operands met drijvende komma worden ondersteund door de numerieke coprocessor 80287.

In figuur 7/3.3-8 zijn de door de 80286 ondersteunde data typen grafisch weergegeven.

I/O Ruimte

De in-/uitgangsruimte bestaat uit 64 k 8-bit of 32 k 16-bit poorten.
De I/O-ruimte wordt geadresseerd met óf een 8-bit poortadres dat in de instructie wordt aangegeven óf door een 16-bit poortadres in het DX-register.
De 8-bit poortadressen worden met nullen verlengd, zodat A15 tot en met A8 LAAG zijn.
De I/O poortadressen 00F8(H) tot en met 00FF(H) zijn gereserveerd.

Interrupties

Door een interruptie wordt de executie overgebracht naar een nieuwe programmaplaats.
Het oude programma-adres (CS:IP) en machine status (Flags) worden op de stack opgeborgen om hervatting van het onderbroken programma mogelijk te maken.

3.3 80286

Function	Interrupt Number	Related Instructions	Does Return Address Point to Instruction Causing Exception?
Divide error exception	0	DIV, IDIV	Yes
Single step interrupt	1	All	
NMI interrupt	2	INT 2 or NMI pin	
Breakpoint interrupt	3	INT 3	
INTO detected overflow exception	4	INTO	No
BOUND range exceeded exception	5	BOUND	Yes
Invalid opcode exception	6	Any undefined opcode	Yes
Processor extension not available exception	7	ESC or WAIT	Yes
Intel reserved-do not use	8-15		
Processor extension error interrupt	16	ESC or WAIT	
Intel reserved-do not use	17-31		
User defined	32-255		

Tabel 7/3.3-8: Toegewezen interrupt-vectoren.

De interrupties worden verdeeld in drie klassen: door hardware geïnitialiseerde, INT instructies en instructie-uitzonderingen. Door hardware geïnitialiseerde interrupties vinden plaats als antwoord op een van buiten afkomstig signaal en worden geklassificeerd als niet-maskeerbaar of maskeerbaar.
Programma's kunnen een interruptie veroorzaken met een INT instructie. Instructie-uitzonderingen vinden plaats als tijdens de poging een instructie uit te voeren een ongewone conditie wordt gedetecteerd, waardoor verdere verwerking van de instructies onmogelijk wordt. Het terugkeeradres door een uitzondering zal altijd naar de instructie wijzen die de uitzondering veroorzaakt en vergezeld gaan van eventuele instructie-prefixen.
Een tabel met maximaal 256 pointers bepaalt de juiste interrupt-serviceroutine voor elke interruptie (tabel 7/3.3-8). De interrupties 0 tot en met 31, waarvan er sommige voor instructie-uitzonderingen worden gebruikt, zijn gereserveerd. Voor elke interruptie moet de 80286 van een 8-bit vector worden voorzien waardoor de tabel op de geschikte plaats wordt betreden. Door uitzonderingen wordt de interrupt-vector intern geleverd. INT instructies bevatten of wijzen naar de vector en maken toegang tot alle 256 interrupties mogelijk.
Door hardware veroorzaakte maskeerbare interrupties leveren de 8-bit vector tijdens een interrupt-acknowledge busreeks aan de CPU. Niet-maskeerbare hardware interrupties gebruiken een van te voren bepaalde intern geleverde vector.

Maskeerbare interrupt (INTR)

De 80286 is voorzien van een pen voor maskeerbare hardware interrupt-request: INTR.
Deze ingang wordt door software vrijgegeven voor gebruik door de interrupt-flagbit (IF) in het flag-woord te setten. Alle 244 door de gebruiker te bepalen interruptiebronnen kunnen deze ingang delen, terwijl zij toch aparte interruptie-afhandelingen behouden.
Een 8-bit vector die door de CPU tijdens de interrupt-acknowledge-reeks wordt ingelezen identificeert de bron van de interruptie (zie Systeem Interface).
Overige maskeerbare interrupties worden gesperd tijdens de behandeling van een interruptie door resetten van de IF, behalve als deel van het antwoord op een uitzonderings-

interruptie. Het opgeborgen flag-woord zal overeenkomen met de enable-status van de processor vóór de interruptie. Totdat het flag-woord is teruggebracht naar het flag-register zal de interrupt-flag nul zijn, tenzij speciaal geset. De interruptie-terugkeerinstructie zorgt ervoor dat het flag-woord, inclusief de originele status van IF, wordt terug geplaatst.

Niet-maskeerbaar interrupt request (NMI)

De 80286 is ook voorzien van een niet-maskeerbare interruptie-ingang (NMI). NMI heeft een hogere prioriteit dan INTR. NMI zou bijvoorbeeld gebruikt kunnen worden voor het aktiveren van een routine bij stroomstoring. De aktivering van deze ingang veroorzaakt een interruptie met een intern geleverde vectorwaarde van 2. Er wordt geen externe interrupt-acknowledge-reeks uitgevoerd.
Terwijl de NMI service-routine wordt uitgevoerd behandelt de 80286 geen andere NMI- of INTR requests en ook geen extentiesegment-overrun-interrupt totdat een interrupt-terugkeerinstructie (IRET) is uitgevoerd of de CPU is gereset. Als een NMI optreedt terwijl al een andere NMI wordt behandeld, wordt de aanwezigheid ervan bewaard om na de eerste IRET instructie te worden uitgevoerd. IF wordt gecleared bij het begin van een NMI interruptie om INTR interrupties tegen te houden.

Single Step Interrupt

De 80286 heeft een interne interruptie die het programma's mogelijk maakt telkens één instructie uit te voeren. Deze "single step interrupt" wordt bestuurd met de single step flag-bit (TF) in het flag-woord. Wanneer deze bit eenmaal is geset treedt een interne single step interruptie op na uitvoering van de volgende instructie. De interruptie maakt de TF-bit schoon en gebruikt een intern geleverde vector van 1. De IRET instructie wordt gebruikt om de TF-bit te setten en de besturing te veranderen waardoor de volgende instructie single step wordt.

Interrupt prioriteiten

Wanneer gelijktijdige interrupt-verzoeken optreden, worden zij in een vaste volgorde verwerkt, zoals in tabel 7/3.3-9 te zien is. Bij de verwerking van de interrupties worden de vlaggen (flags) en het terugkeeradres bewaard en wordt CS:IP geset om naar de eerste instructie van de interrupt-handler te wijzen. Indien andere interrupties zijn vrijgegeven (enabled), worden die verwerkt voordat de eerste instructie van de lopende interrupt-handler wordt uitgevoerd. De laatst verwerkte interruptie is daardoor de eerste die behandeld wordt.

Order	Interrupt
1	Instruction exception
2	Single step
3	NMI
4	Processor extension segment overrun
5	INTR
6	INT instruction

Tabel 7/3.3-9: Prioriteiten (volgorde van afhandeling van de interrupties).

Initialisatie en Processor Reset

Initialisatie (opstarten) van de processor geschiedt door de RESET-ingang HOOG te maken. RESET dwingt de 80286 alle verwerkingen en aktiviteiten op de lokale bus te stoppen. Zolang RESET aktief is zal geen enkele instructie of busaktiviteit plaatsvinden. Nadat RESET weer niet-aktief is geworden en een interne verwerkingsinterval is verstreken begint de 80286 in de reële adresseermode met de instructie op de lokatie FFFFF0(H). Door RESET worden ook enkele registers met van te voren vastgestelde waarden gevuld, die in tabel 7/3.3-10 worden getoond.
HOLD mag niet aktief zijn gedurende de tijd vanaf de voorflank van RESET tot 34 klokperioden na de achterflank van RESET.

3.3 80286

Flag word	0002(H)
Machine Status Word	FFF0(H)
Instruction pointer	FFF0(H)
Code segment	F000(H)
Data segment	0000(H)
Extra segment	0000(H)
Stack segment	0000(H)

Tabel 7/3.3-10: De begin-inhouden van de registers in een 80286 na het resetten.

Bit Position	Name	Function
0	PE	Protected mode enable places the 80286 into protected mode and cannot be cleared except by RESET.
1	MP	Monitor processor extension allows WAIT instructions to cause a processor extension not present exception (number 7).
2	EM	Emulate processor extension causes a processor extension not present exception (number 7) on ESC instructions to allow emulating a processor extension.
3	TS	Task switched indicates the next instruction using a processor extension will cause exception 7, allowing software to test whether the current processor extension context belongs to the current task.

Tabel 7/3.3-11: Bit-functies van het MSW.

Beschrijving van het Machine Status Woord

Met het machine statuswoord (MSW) wordt bijgehouden wanneer een taakwisseling plaats vindt en wordt de bedrijfsmode van de 80286 geregeld.
Het is een 16-bit register waarvan alleen de laagste vier bits worden gebruikt (zie ook figuur 7/3.3-5).
Eén bit zet de processor in de beveiligde (protected) mode, terwijl de andere drie bits de uitbreidings-interface van de processor besturen (tabel 7/3.3-11). Na resetten is de inhoud van dit register FFF0(H), waardoor de 80286 in de reële adresseringsmode staat.

Met behulp van de LMSW en SMSW instructies kan het Machine Statuswoord (MSW) in de reële adresseringsmode worden geladen en opgeborgen. In tabel 7/3.3-12 is te zien hoe TS, EM en MP het best kunnen worden gebruikt.

Halt

De HLT instructie laat de uitvoering van een programma stoppen en voorkomt dat de CPU de lokale bus gebruikt voordat opnieuw gestart is.
Door NMI, INTR (met IF = 1) of RESET wordt de 80286 uit de haltpositie gedwongen. Als er geïnterrumpeerd is wijst de bewaarde CS:IP naar de eerstvolgende instructie na HLT.

TS	MP	EM	Recommended Use	Instructions Causing Exception 7
0	0	0	Initial encoding after RESET. 80286 operation is identical to 8086, 88.	None
0	0	1	No processor extension is available. Software will emulate its function.	ESC
1	0	1	No processor extension is available. Software will emulate its function. The current processor extension context may belong to another task.	ESC
0	1	0	A processor extension exists.	None
1	1	0	A processor extension exists. The current processor extension context may belong to another task. The Exception 7 on WAIT allows software to test for an error pending from a previous processor extension operation.	ESC or WAIT

Tabel 7/3.3-12: Aanbevolen MSW-coderingen voor Processor Extension Control.

De reële 8086 adresseringsmode

Inleiding

De 80286 voert in de reële adresseringsmode (real address mode) een volledig opwaarts compatibele superset van de 8086 instructieset uit. In de reële adresseringsmode is de 80286 object-code compatibel met software voor de 8086/8088. De architectuur voor de reële adresseringsmode (registers en adresseermodes) wordt precies beschreven in het gedeelte Basis Architectuur van de 80286.

Afmetingen van het geheugen

Het fysieke werkgeheugen is een aansluitende reeks van maximaal 1.048.576 bytes (één megabyte), adresseerbaar met de pennen A0 tot en met A19 en $\overline{BHE}$. A20 tot en met A23 moeten worden genegeerd.

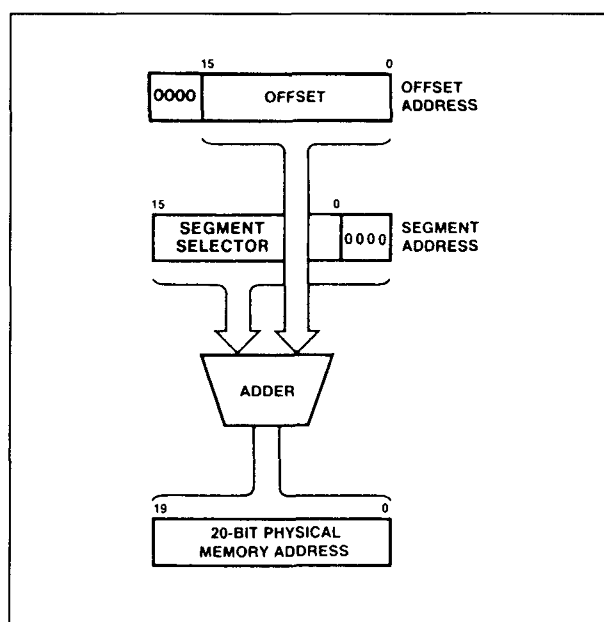
Geheugen-adressering

In de reële adresseringsmode is het fysieke geheugen is één aansluitende reeks van maximaal 1.048.576 bytes. Deze 1 MB kan worden geadresseerd met de pennen A0 tot en met A19 plus $\overline{BHE}$. In de reële mode kan het voorkomen dat A20 tot en met A23 niet altijd nul zijn. A20 tot en met A23 moeten dan ook niet worden gebruikt als de 80286 in de reële mode werkt.

Het selectorgedeelte van een pointer wordt beschouwd als de hoogste 16 bits van een 20-bit segment-adres. De laagste 4 bits van het 20-bit segment-adres zijn altijd nul. Segment-adressen beginnen daardoor altijd op veelvouden van 16 bytes (zie ook figuur 7/3.3-9).

In de reële adresseringsmode zijn alle segmenten 64 kB groot en kunnen worden gelezen, beschreven of uitgevoerd. Indien data-operands of instructies proberen over de rand van een segment te gaan (dat wil zeggen dat van een woord de laagste byte op offset FFFF(H) ligt en de hoogste byte op offset 0000(H)), kan een uitzondering of interruptie optreden.

Als in de reële adresseringsmode voor de informatie niet de gehele 64 kB wordt gebruikt, mag in het niet gebruikte einde van het segment overlay plaatsvinden door een ander segment om de aan het geheugen gestelde eisen te verminderen.



Figuur 7/3.3-9: Grafische voorstelling van de adressering in de 8086 reële mode.

Gereserveerde geheugenplaatsen

De 80286 reserveert in de reële adresseringsmode twee vaste geheugengebieden (figuur 7/3.3-10), namelijk het systeem-initialisatiegebied en het interrupt-tabelgebied. De lokaties van adres FFFF0(H) tot en met adres FFFFF(H) zijn bestemd voor de initialisatie van het systeem. De initiële executie begint op adres FFFF0(H). De plaatsen 00000(H) tot en met 003FF(H) zijn gereserveerd voor interrupt-vectoren.

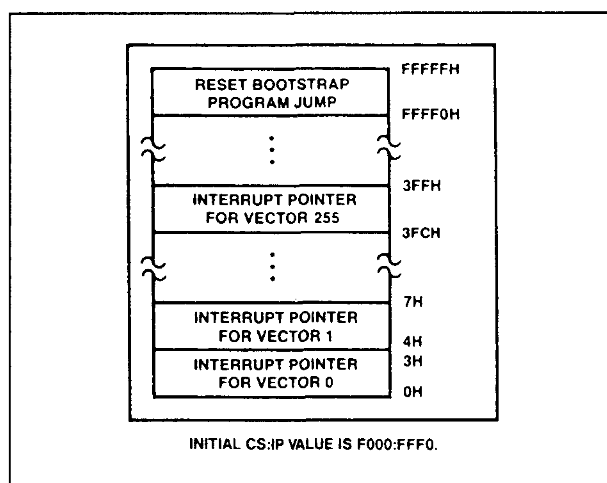
Interrupties

In tabel 7/3.3-13 zijn de interrupt-vectoren te zien die zijn gereserveerd voor uitzonderingen en interrupties die een adresseringsfout aangeven.

3.3 80286

Function	Interrupt Number	Related Instructions	Return Address Before Instruction?
Interrupt table limit too small exception	8	INT vector is not within table limit	Yes
Processor extension segment overrun interrupt	9	ESC with memory operand extending beyond offset FFFF(H)	No
Segment overrun exception	13	Word memory reference with offset = FFFF(H) or an attempt to execute past the end of a segment	Yes

Tabel 7/3.3-13: Interrupties door adresseerfouten in de reële adresseringsmode.



Figuur 7/3.3-10: Initieel gereserveerde geheugenplaatsen in de 8086 reële adresseringsmode.

De uitzonderingen laten de CPU in de toestand die bestond voordat werd geprobeerd de falende instructie uit te voeren (behalve voor PUSH, POP, PUSHA of POPA).

Initialisatie in de beschermde mode

Om de 80286 voor te bereiden op de beschermde mode wordt de LIDT instructie gebruikt om de basis van de 24-bit interruptietabel en de 16-bit limiet voor de interruptietabel van de beschermde mode te laden. Deze instructie kan ook een basis en een limiet voor de interrupt-vectortabel in de reële mode zetten. Na resetten wordt de basis van de interruptietabel op 000000(H) geïnitialiseerd en de afmeting op 03FF(H) gezet. Deze waarden komen overeen met 8086/8088 software. LIDT mag alleen worden uitgevoerd ter voorbereiding van de beschermde mode.

Shutdown

Shutdown treedt op wanneer een ernstige fout wordt gedetecteerd waardoor verdere bewerking door de CPU onmogelijk wordt. Shutdown en halt worden extern gesignaleerd via een halt bus operatie. Zij kunnen worden onderscheiden door A1 (HOOG = halt, LAAG = shutdown).
In de reële adresseringsmode kan shutdown onder twee omstandigheden optreden:
- wanneer uitzondering 8 of 13 gebeurt en de interrupt-vector niet binnen de IDT limiet valt;
- wanneer een CALL INT of PUSH instructie probeert rond het stack-segment te gaan wanneer SP oneven is.

Een NMI-signaal kan de CPU uit shutdown brengen als de IDT-limiet tenminste 000F(H) en SP groter dan 0005(H) zijn. In andere gevallen kan shutdown alleen worden verlaten via de RESET-ingang.

Beschermde virtuele adresseringsmode

Inleiding

De 80286 voert een volledig opwaarts compatibele superset van de 8086 instructieset uit in de beschermde virtuele adresserings-

mode (protected mode). In de beschermde mode zijn ook geheugen-managent en protectiemechanismen en aanverwante instructies mogelijk.

De 80286 komt vanuit de reële adresseringsmode in de beschermde virtuele adresseringsmode door setten van de PE-bit (Protection Enable) in het machine statuswoord met de Load Machine Status Word (LMSW) instructie. In de beschermde mode worden uitgebreide fysieke en virtuele geheugen-adresseringsruimte, beschermingsmechanismen voor het geheugen en nieuwe operaties voor de ondersteuning van systemen en virtueel geheugen geboden.

Alle registers, instructies en adresseringsmodes die in het gedeelte over de 80286 basis architectuur werden beschreven blijven gelijk. Programma's voor de 8086, 8088, 80186 en de reële adresseringsmode 80286 kunnen ook in de beschermde mode draaien, ingebedde constanten voor segmentsectoren zijn echter verschillend.

Geheugen-afmetingen

De beschermde mode 80286 levert 1 gigabyte adresruimte per taak, onderverdeeld in 16 megabyte fysieke adresruimte die wordt gedefinieerd door de adrespennen A23 tot en met A0 en $\overline{BHE}$.

De virtuele adresruimte mag groter zijn dan de fysieke adresruimte, aangezien gebruik van een adres dat niet in een fysieke adreslokatie past een herstartbare uitzondering oplevert.

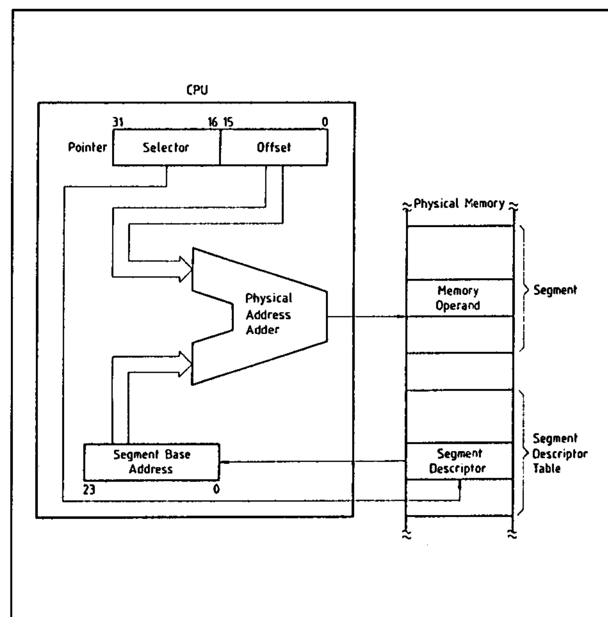
Geheugen-adressering

Net als bij de reële adresseringsmode gebruikt de beschermde mode 32-bit pointers die uit 16-bit selector en offset-componenten bestaan.

De selector specificeert echter een index naar een zich in het geheugen bevindende tabel, in plaats van de hoogste 16 bits van een reëel geheugenadres.

Het 24-bit basisadres van het gewenste segment wordt verkregen uit de tabellen in het geheugen.

De 16-bit offset wordt aan het basisadres van het segment toegevoegd om zodoende het fysieke adres te vormen (zie ook figuur 7/3.3-11). Telkens wanneer een segment-register met een selector wordt geladen wordt automatisch door de CPU naar de tabellen verwezen. Alle 80286 instructies die een segment-register laden verwijzen naar de in het geheugen geplaatste tabellen zonder dat daar extra software voor nodig is. De tabellen bevatten 8-byte waarden die trefwoorden (descriptoren) genoemd worden.



Figuur 7/3.3-11: Geheugen-adressering in de beschermde mode.

Trefwoorden (descriptoren)

Het geheugengebruik wordt door descriptoren bepaald. Speciale typen descriptoren bepalen ook nieuwe functies voor besturingsoverdracht en taakwisseling. De 80286 heeft segment-descriptoren voor code, stack en data-segmenten en systeem besturingsdescriptoren voor speciale systeem-datasegmenten en besturingsoverdracht-operaties. Toegang tot de descriptoren wordt verkregen in de vorm van geblokkeerde bus-operaties om de integriteit van de descriptoren in multi-processor systemen te garanderen.

3.3 80286

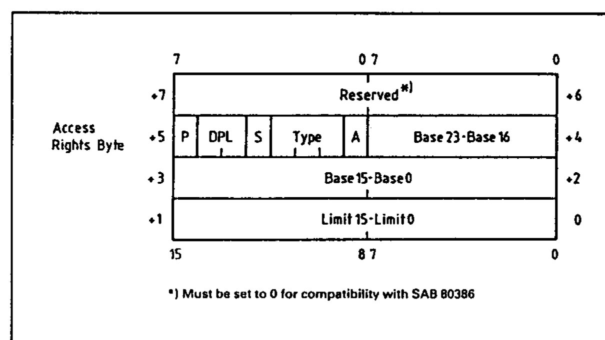
	Bit Position	Name		Function	
	7	Present (P)	P = 1	Segment is mapped into physical memory	
			P = 0	No mapping to physical memory exists, base and limit are not used	
	6-5	Descriptor privilege level (DPL)		Segment privilege attribute used in privilege tests	
	4	Segment descriptor (S)	S = 1	Code or data (includes stacks) segment descriptor	
			S = 0	System segment descriptor or gate descriptor	
Type field definition	3	Executable (E)	E = 0	Data segment descriptor type is:	If data segment (S = 1, E = 0)
	2	Expansion direction (ED)	ED = 0	Expand up segment, offsets must be ≤ limit	
			ED = 1	Expand down segment, offsets must be > limit	
	1	Writeable (W)	W = 0	Data segment may not be written into	
			W = 1	Data segment may be written into	
	3	Executable (E)	E = 1	Code segment descriptor type is:	If code segment (S = 1, E = 1)
	2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged	
	1	Readable (R)	R = 0	Code segment may not be read	
			R = 1	Code segment may be read	
	0	Accessed (A)	A = 0	Segment has not be accessed	
			A = 1	Segment selector has been loaded into segment register or used by selector test instructions	

Tabel 7/3.3-14: Definitie van de toegangsrechtenbyte (Access Rights Byte).

Code- en data-segment descriptoren (S = 1)

Naast segment-basisadressen, code en data bevatten descriptoren andere segment-attributen, zoals segment-afmetingen (1 tot 64 kB), toegangsrechten (read-only, read/write, execute only en execute/read) en aanwezigheid in het geheugen (voor virtuele geheugensystemen). In figuur 7/3.3-12 is de code- of data-segment descriptor te zien, met de beschrijving van de bits in de toegangsrechtenbyte in tabel 7/3.3-14.
Wanneer een segment op een andere manier wordt gebruikt dan door de segment-descriptor is aangegeven, wordt de geheugencyclus tegengehouden en een uitzondering of interruptie veroorzaakt.

Code en data (inclusief stack data) worden opgeborgen in twee soorten segmenten: code-segmenten en data-segmenten. Beide soorten worden geïdentificeerd en beschreven door segment-descriptoren (S = 1). Code-segmenten worden geïdentificeerd door het op 1 zetten van de executable bit (E) in de toegangsrechtenbyte van de descriptor. De toegangsrechtenbyte van zowel code- als data-segment descriptortypen hebben drie gemeenschappelijke velden: Present (P) bit, Descriptor Privilege Level (DPL) en Accessed (A) bit.



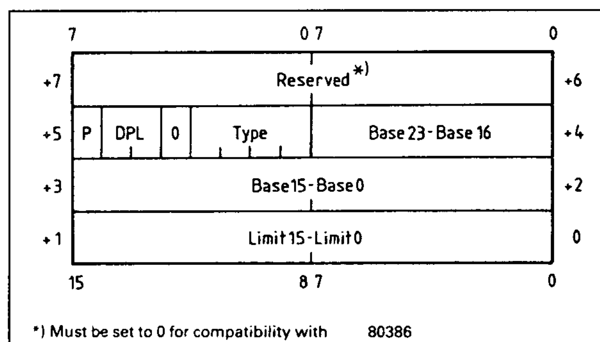
Figuur 7/3.3-12: Code- of data-segment descriptor.

Als P = 0 zal elke poging dit segment te gebruiken een niet-aanwezig (not-present) uitzondering veroorzaken. Met DPL wordt het privilege-niveau van de segment-descriptor gespecificeerd. DPL regelt wanneer de descriptor door een taak gebruikt mag worden. De A-bit laat zien of het segment eerder werd betreden voor profilering

van het gebruik (noodzakelijk voor virtuele geheugensystemen). De CPU zal deze bit altijd setten bij toegang tot de descriptor.
Data-segmenten (S = 1, E = 0) kunnen, onder besturing van de W-bit van de toegangsrechtenbyte, read-only of read/write zijn. Read-only (W = 0) data-segmenten mogen niet worden beschreven. Data-segmenten kunnen, zoals bepaald wordt door de Expansion Direction (ED) bit, in twee richtingen groeien: opwaarts (ED = 0) voor data-segmenten en neerwaarts (ED = 1) voor een segment dat een stack bevat. Het limietveld voor een data-segment descriptor wordt, afhankelijk van de ED-bit verschillend geïnterpreteerd (zie tabel 7/3.3-14).
Een code-segment (S = 1, E = 1) kan execute-only of execute/read zijn, afhankelijk van de Readable (R) bit. Code-segmenten mogen nooit worden beschreven en execute-only code-segmenten (R = 0) mogen niet worden uitgelezen. Een code-segment kan ook een attribuut bevatten dat "conforming" (C) heet. Programma's die op verschillende privilege-niveaus werken, kunnen een conformerend code-segment met elkaar delen. De DPL van een conformerend code-segment bepaalt het bereik van privilege-niveaus waarop het segment kan worden uitgevoerd (zie het verderop beschreven gedeelte over privileges).
Het limietveld identificeert de laatste byte van een code-segment.

Systeem-segment descriptoren (S = 0, type = 1 - 3)

Naast code- en data-segment descriptoren kent de beschermde mode van de 80286 systeem-segment descriptoren. Deze descriptoren definiëren speciale systeem data-segmenten die een tabel met descriptoren bevatten (Local Descriptor Table Descriptor) of segmenten die de executie-status van een taak bevatten (Task State Segment Descriptor).
Figuur 7/3.3-13 toont de formaten voor de speciale systeem data-segment descriptoren. De descriptoren bevatten een 24-bit basisadres van het segment en een 16-bit limiet. De toegangsbyte definieert het type descriptor en de status en privilege-niveau ervan. De inhouden van de descriptor zijn geldig en het segment bevindt zich in fysiek geheugen als P = 1. Als P = 0 is het segment niet geldig. Het DPL-veld wordt alleen gebruikt in Task State Segment descriptoren en geeft een indicatie van het privilege-niveau waarop de descriptor gebruikt mag worden. Aangezien de Local Descriptor Table descriptor alleen mag worden gebruikt door een speciale bevoorrechte instructie, wordt het DPL-veld niet gebruikt. Bit 4 van de toegangsbyte is 0 om aan te geven dat er sprake is van een systeem-besturings descriptor. Het type-veld specificeert de soort descriptor zoals aangegeven in tabel 7/3.3-15.



Figuur 7/3.3-13: Systeem-segment descriptor.

Gate descriptoren (S = 0, type = 4 - 7)

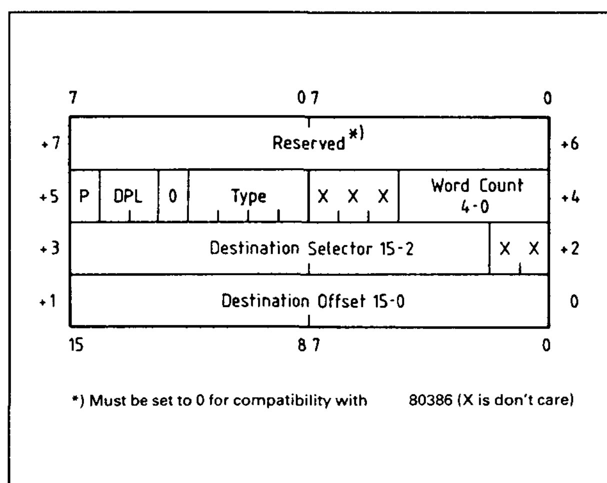
De toegang tot ingangspunten binnen het doel code-segment wordt bestuurd met poorten (gates).
De gate-descriptoren zijn:
- call gates;
- task gates;
- interrupt gates;
- trap gates.

Door gates is een mate van omleiding tussen de bron en de bestemming van een besturingsoverdracht mogelijk. Deze omleiding stelt de CPU in staat automatisch beveiligingschecks uit te voeren en het ingangspunt van de bestemming te regelen.

3.3 80286

Name	Value	Description
TYPE	1 2 3	Available task state segment Local descriptor table descriptor Busy task state segment
P	0 1	Descriptor contents are not valid Descriptor contents are valid
DPL	0–3	Descriptor privilege level
BASE	24-bit number	Base address of special system data segment in real memory
LIMIT	16-bit number	Offset of last byte in segment

Tabel 7/3.3-15: Formaten van de systeem-segment descriptor.



Figuur 7/3.3-14: Gate descriptor.

Call gates worden gebruikt om privilege-niveaus te veranderen, task gates voeren taakwisselingen uit en interrupt en trap gates dienen om interrupt service-routines te specificeren. De interrupt gate spert interrupties (reset IF), de trap gate niet.

In tabel 7/3.3-16 is het formaat van de gate descriptoren te zien. De descriptor bevat een bestemmings (destination) pointer die naar de descriptor van het doelsegment en de offset van het ingangspunt wijst.
De destination selector in een interrupt gate, trap gate en call gate moet een relatie hebben tot een code-segment descriptor.

Deze gate descriptoren bevatten het ingangspunt om te voorkomen dat een programma een illegaal ingangspunt opwekt en gebruikt. Task gates mogen alleen betrekking hebben op een taak-status segment. Aangezien task gates een taakwisseling oproepen, wordt de bestemmings-offset niet gebruikt bij de task gate.

Als een bestemmings selector niet op het juiste descriptortype slaat, wordt uitzondering 13 gegenereerd. Het word-count veld in de call gate descriptor wordt gebruikt om, wanneer een besturingsoverdracht privilege-niveaus verandert, het aantal parameters aan te geven (0 tot 31 woorden) dat automatisch uit de stack van de oproeper naar de stack van de opgeroepen routine moet worden gekopieerd. Het word-count veld wordt door geen enkele andere gate descriptor gebruikt. Het toegangsbyte formaat is voor alle gate descriptoren hetzelfde. P = 1 geeft aan dat de inhouden van de gates geldig zijn. P = 0 (inhouden niet geldig) veroorzaakt uitzondering 11, indien verwezen. DPL is het privilege-niveau van de descriptor en specificeert wanneer deze descriptor door een taak mag worden gebruikt. Bit 4 moet 0 zijn om een systeembesturings-descriptor aan te geven. Het type-veld specificeert de soort descriptor, zoals in tabel 7/3.3-16 is aangegeven.

Name	Value	Description
TYPE	4 5 6 7	– Call gate – Task gate – Interrupt gate – Trap gate
P	0 1	– Descriptor contents are not valid – Descriptor contents are valid
DPL	0–3	Descriptor privilege level
WORD COUNT	0–31	Number of words to copy from callers stack to called procedures stack. Only used with call gate
DESTINATION SELECTOR	16-bit selector	Selector to the target code segment (call, interrupt or trap gate)
DESTINATION OFFSET	16-bit offset	Entry point within the target code segment

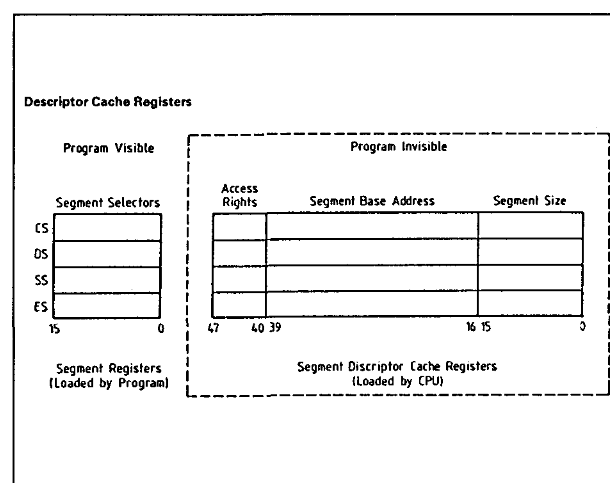
Tabel 7/3.3-16: Formaten van de gate descriptor.

Segment descriptor cache-registers

Aan elk van de vier segment-registers (CS, SS, DS, ES) wordt een segment descriptor cache-register toegewezen. Segment descriptoren worden automatisch naar een segment descriptor cache-register overgebracht (cached) wanneer het betrokken segment-register met een selector wordt geladen (figuur 7/3.3-15). Segment descriptor cache-registers mogen alleen met segment-descriptoren worden geladen. Zijn die eenmaal geladen, dan gebruiken alle verwijzingen naar dat geheugen-segment de in cache opgeslagen descriptor-informatie in plaats van telkens opnieuw de descriptor te benaderen. De descriptor cache-registers zijn voor programma's niet zichtbaar. Er bestaan geen instructies om hun inhouden op te slaan en zij veranderen alleen wanneer een segment-register wordt geladen.

Selectorvelden

Een beschermde mode selector heeft drie velden: descriptor entry index, locale of globale descriptor-tabelindicator (TI) en selector privilege (RPL), zoals figuur 7/3.3-16 laat zien. Deze velden selecteren één van de twee in geheugen gebaseerde descriptor-tabellen, selecteren de juiste ingang en maken testen op hoge snelheid van het selector privilege-attribuut mogelijk.



Figuur 7/3.3-15: Descriptor cache-registers.

Lokale en globale descriptor-tabellen

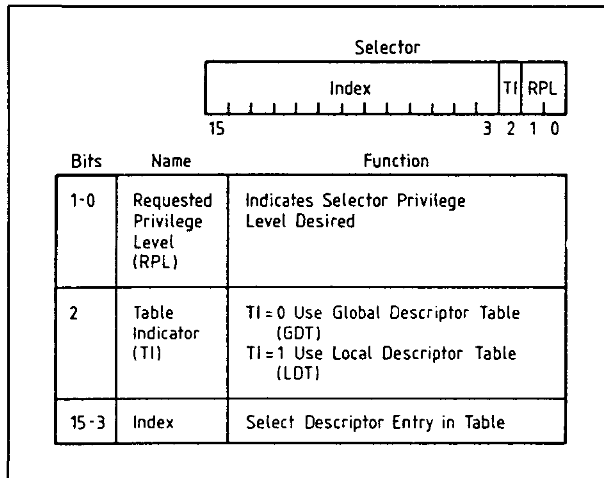
- Twee tabellen

  De descriptor-tabellen bevatten alle descriptoren waartoe een taak op elk willekeurig moment toegang kan krijgen. Een descriptor-tabel is een lineair array van maximaal 8.192 descriptoren. De hoogste 13 bits van de selector-waarde zijn een index tot een descriptor-tabel. Elke tabel heeft een 24-bit basis-register om de descriptor-tabel in fysiek geheugen te lokaliseren en een 16-bit limiet-register dat descriptor-toegangen beperkt houdt tot de gedefinieerde limieten van de tabel (zie

3.3 80286

figuur 7/3.3-17). Een herstartbare uitzondering (13) treedt op als een poging wordt gedaan om met een descriptor buiten de grenzen van de tabel te werken.

- Eén tabel
  De Globale Descriptor-tabel (GDT) bevat descriptoren die door alle taken mogen worden gebruikt. De andere tabel, de Lokale Descriptor-tabel (LDT) bevat descriptoren die voor een taak privé kunnen zijn. Elke taak mag zijn eigen privé LDT hebben. De GDT kan, op interrupt en trapdescriptoren na, alle descriptor-typen bevatten. De LDT kan alleen segment, task gate en call gate descriptoren bevatten.
  Een segment kan niet door een taak worden bereikt als de segment-descriptor ervan in geen van beide descriptor-tabellen is opgenomen.



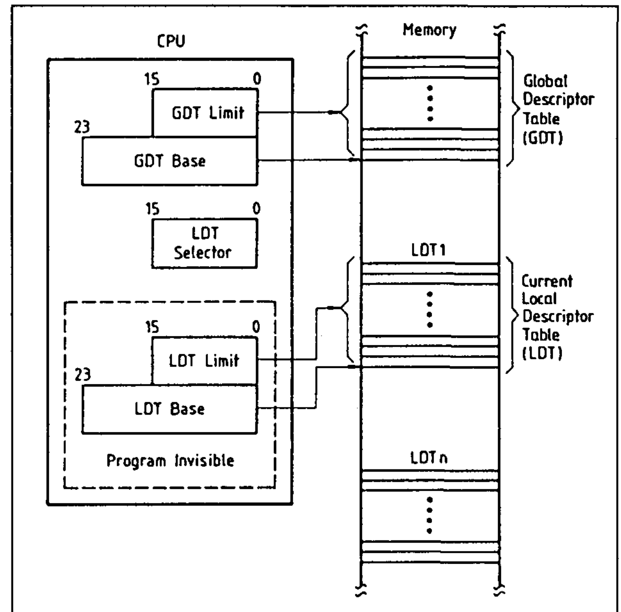
Bits	Name	Function
1-0	Requested Privilege Level (RPL)	Indicates Selector Privilege Level Desired
2	Table Indicator (TI)	TI = 0 Use Global Descriptor Table (GDT) TI = 1 Use Local Descriptor Table (LDT)
15-3	Index	Select Descriptor Entry in Table

Figuur 7/3.3-16: Selectorvelden.

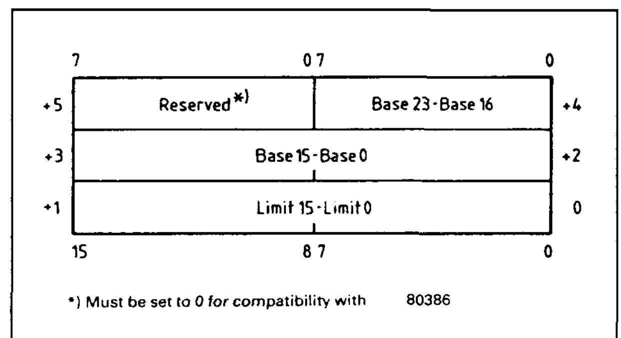
De LGDT en LLDT instructies laden de basis en de limiet van de lokale en globale descriptor-tabellen. Het zijn geprivilegeerde instructies, hetgeen wil zeggen dat ze slechts mogen worden uitgevoerd door vertrouwde programma's op niveau 0.

De LGDT-instructie laadt een 6-byte veld, bestaande uit de 16-bit tabel-limiet en 24-bit fysiek basisadres van de Globale Descriptor-tabel, zoals in figuur 7/3.3-18 te zien is.

De LLDT-instructie laadt een selector die verwijst naar een Lokale Descriptor-tabel die het basisadres en de limiet voor een LDT bevat (zie ook tabel 7/3.3-15).



Figuur 7/3.3-17: Lokale en globale descriptor-tabel definities.

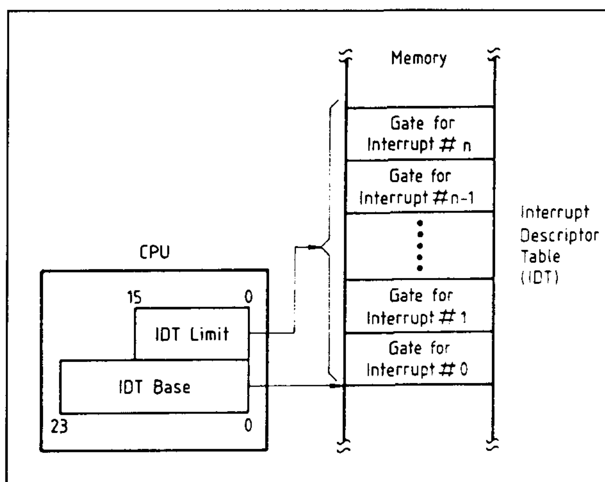


Figuur 7/3.3-18: Globale descriptor-tabel en data-type van de Interrupt descriptor-tabel.

Interrupt descriptor-tabel

De beschermde mode van de 80286 heeft nog een derde descriptor-tabel, de Interrupt Descriptor tabel (IDT) genaamd, die wordt gebruikt om maximaal 256 interrupties te definiëren (figuur 7/3.3-19). Hij kan alleen

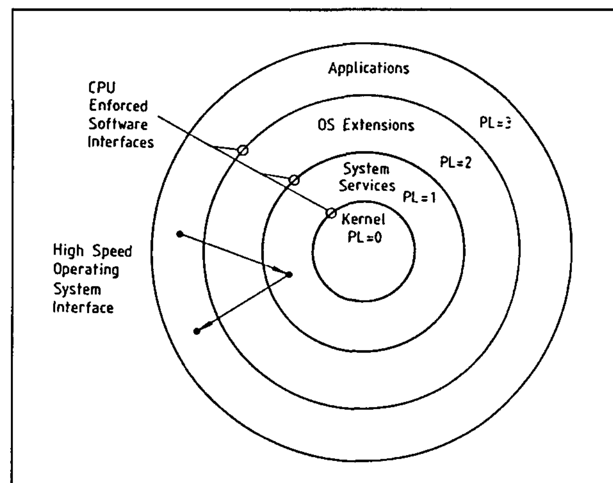
task gates, interrupt gates en trap gates bevatten. De IDT (Interrupt Descriptor Tabel) heeft een 24-bit fysieke basis en een 16-bit limiet-register in de CPU. De geprivilegeerde LIDT instructie laadt deze registers met een 6-byte waarde van identieke vorm naar die van de LGDT instructie (figuur 7/3.3-18 en Initialisatie van de Beschermde Mode).
Verwijzingen naar IDT-ingangen worden gemaakt via INT instructies, externe interruptvectoren of uitzonderingen. De IDT moet tenminste 256 bytes groot zijn voor het toewijzen van ruimte aan alle gereserveerde interrupties.



Figuur 7/3.3-19: Definitie van de Interrupt Descriptor-tabel.

Privilege

De 80286 heeft een hiërarchisch privilegesysteem op vier niveaus dat het gebruik van geprivilegeerde instructies en de toegang tot descriptoren (en de bijbehorende segmenten) binnen een taak regelt. Het privilege op vier niveaus, zoals figuur 7/3.3-20 toont, is een uitbreiding op de user/supervisor mode bij minicomputers. De privilege-niveaus zijn van 0 tot 3 genummerd, waarbij niveau 0 het meest bevoorrecht is. Door de privilegeniveaus wordt voorzien in beveiliging binnen een taak. Taken worden geïsoleerd door elke taak een eigen LDT te geven. Bedrijfssysteem-routines, interrupt handlers en andere systeem-software kunnen binnen de virtuele adresruimte van een taak worden inbegrepen en beveiligd door gebruik te maken van de vier privilege-niveaus. Elke taak in het systeem heeft een aparte stack voor elk van zijn privilege-niveaus.



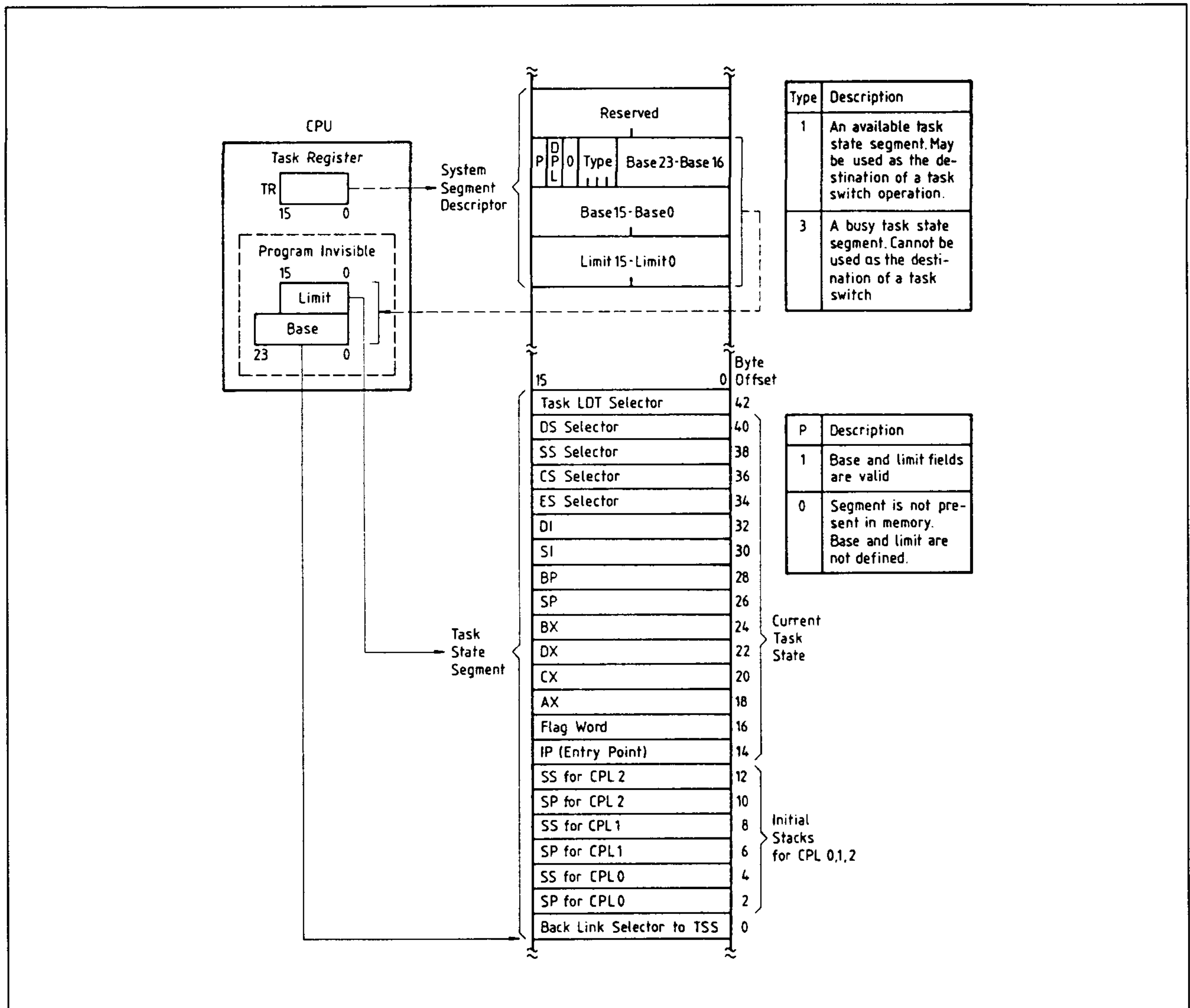
Figuur 7/3.3-20: Hiërarchische privilege-niveaus.

Taken, descriptoren en selectoren hebben een privilege-niveau attribuut die uitmaakt of de descriptor gebruikt mag worden. Door taken te voorzien van privileges wordt het gebruik van instructies en descriptoren effectiever. Descriptor- en selector-privilege hebben alleen invloed op de toegang tot de descriptor.

Taak-privilege

Een taak wordt altijd op één van de vier privilege-niveaus uitgevoerd. Het privilegeniveau van een taak op een willekeurig moment wordt Current Privilege Level (CPL) genoemd en wordt bepaald door de laagste twee bits van het CS-register.
CPL kan tijdens uitvoering in een enkel codesegment niet veranderen. De CPL van een taak kan slechts worden veranderd door besturingsoverdrachten via gate-descriptoren naar een nieuw code-segment (zie Control Transfer).

3.3 80286



Figuur 7/3.3-21: Task State Segment en TSS Registers.

Taken worden uitgevoerd op de CPL-waarde die door de code-segment selector binnen TSS wordt gespecificeerd, wanneer de taak via een taakwissel-operatie wordt geïnitialiseerd (zie figuur 7/3.3-21). Een taak die wordt uitgevoerd op niveau 0 heeft toegang tot alle data-segmenten die in de GDT en de LDT van de taak zijn gedefinieerd. Een taak die op niveau 3 wordt uitgevoerd heeft de meest beperkte toegang tot data en wordt beschouwd als het minst vertrouwde niveau.

Descriptor-privilege

Het discriptor-privilege wordt gespecificeerd door het Descriptor Privilege Level (DPL) veld van de descriptor toegangsbyte. DPL specificeert van de minst vertrouwde taak het privilege-niveau (CPL) waarop een taak toegang mag hebben tot de descriptor.
Descriptoren met DPL = 0 hebben de meeste bescherming en mogen alleen worden benaderd door taken die op privilege-niveau 0 (CPL = 0) worden uitgevoerd.

3.3 80286

Descriptoren met DPL = 3 zijn het slechts beveiligd, aangezien taken met CPL = 0, 1, 2 of 3 toegang kunnen verkrijgen.
Deze regel geldt voor alle descriptoren, behalve voor LDT-descriptoren.

Selector-privilege
Selector-privilege wordt gespecificeerd door het Request Privilege Level (RPL) veld in de twee minst belangrijke bits van een selector. Selector RPL kan voor het gebruik van een selector een minder vertrouwd privilege-niveau vaststellen dan de CPL. Dit niveau wordt het effectieve privilege-niveau (EPL) van een taak genoemd. RPL kan met deze selector slechts het toegangsgebied van een taak tot data verminderen. Het effectieve privilege van een taak is het numerieke maximum van RPL en CPL. Een selector met RPL = 0 voegt geen extra beperking toe op het gebruik ervan, terwijl een selector met RPL = 3, onafhankelijk van de CPL van een taak, slechts betrekking kan hebben op privilege-niveau 3. RPL wordt meestal gebruikt om te verifiëren dat pointer-parameters die aan een procedure zijn toegewezen geen data mogen gebruiken op een niveau dat meer vertrouwd is dan dat van de gebruiker.

Descriptor-toegang en privilege-waardering
Bij het vaststellen van de mogelijkheid van een taak om toegang te krijgen tot een segment moet rekening worden gehouden met het type segment dat wordt betreden, de gebruikte instructie en CPL, RPL en DPL. De twee basis-typen segment-toegang zijn besturingsoverdracht (control transfer: selectoren die in CS worden geladen) en data (selectoren die in DS, ES of SS worden geladen).

Toegang tot data-segmenten
Instructies die selectoren in DS en ES laden moeten betrekking hebben op een data-segment descriptor of een uitleesbare code-segment descriptor. De CPL van de taak en de RPL van de selector moeten hetzelfde of een hoger privilege hebben dan de descriptor DPL (numeriek gelijk of lager). Over het algemeen kan een taak alleen data-segmenten bereiken op dezelfde of lager geprivilegeerde niveaus dan de CPL of RPL (welk van de twee numeriek hoger is) om te voorkomen dat een programma verboden data gebruikt. Een uitzondering op de regel is een uitleesbaar conformerend codesegment. Dit type code-segment kan op elk privilege-niveau worden gelezen. Als de privilege-checks falen (bijvoorbeeld als DPL numeriek minder is dan het maximum van CPL en RPL) of een onjuist type descriptor wordt gebruikt (bijvoorbeeld een gatedescriptor of een excute only code-segment) treedt uitzondering 13 op.
Als het segment niet aanwezig is, wordt uitzondering 11 gegenereerd. Instructies die selectoren in SS laden moeten betrekking hebben op data-segment descriptoren voor beschrijfbare data-segmenten. Het descriptor-privilege (DPL) en RPL moeten gelijk zijn aan CPL. Alle andere descriptor-typen of een overtreding van het privilege-niveau veroorzaken uitzondering 13. Een "not-present" fout levert uitzondering 12 op.

Besturingsoverdracht (Control transfer)
Er kunnen vier typen besturingsoverdracht plaatsvinden wanneer door een control transfer operatie een selector in CS wordt geladen (zie tabel 7/3.3-17). Elk type overdracht kan slechts optreden als de operatie die de selector laadde betrekking heeft op het juiste descriptor-type. Overtredingen van deze regels voor descriptor-gebruik (bijvoorbeeld JMP door een call gate of RET naar een Task State Segment) veroorzaken uitzondering 13. De mogelijkheid om een descriptor aan te wijzen voor besturingsoverdracht is ook onderworpen aan privilege-regels. Een CALL of JUMP instructie mag alleen betrekking hebben op een codesegment descriptor met DPL gelijk aan de CPL van de taak of een conformerend segment met DPL van een gelijk of hoger privilege dan CPL.

3.3 80286

Control Transfer Types	Operation Types	Descriptor Referenced	Descriptor Table
Intersegment within the same privilege level	JMP, CALL, RET, IRET*	Code Segment	GDT/LDT
Intersegment to the same or higher privilege level Interrupt within task may change CPL.	CALL	Call Gate	GDT/LDT
	Interrupt Instruction, Exception, External Interrupt	Trap or Interrupt Gate	IDT
Intersegment to a lower privilege level (changes task CPL)	RET, IRET*	Code Segment	GDT/LDT
	CALL, JMP	Task State Segment	GDT
Task Switch	CALL, JMP	Task Gate	GDT/LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

*NT (Nested Task bit of flag word) = 0
**NT (Nested Task bit of flag word) = 1

Tabel 7/3.3-17: Descriptor-typen die voor besturingsoverdracht worden gebruikt.

De RPL van de selector die wordt gebruikt om de code-descriptor aan te wijzen moet evenveel privilege hebben als CPL.

RET en IRET instructies mogen slechts betrekking hebben op code-segment descriptoren met een descriptor-privilege gelijk aan of lager dan de CPL van de taak. De selector die in CS is geladen is het terugkeeradres van de stack. Na terugkeer is RPL van de selector de nieuwe CPL van de taak. Als CPL verandert wordt de oude stackpointer gepopt na het terugkeeradres. Wanneer een JMP of CALL betrekking heeft op een Task State Segment descriptor moet DPL van de descriptor gelijk of minder geprivilegeerd zijn dan de CPL van de taak. Verwijzing naar een geldige Task State Segment descriptor veroorzaakt een taakwisseling. Verwijzing naar een Task State Segment descriptor op een meer bevoorrecht niveau dan de CPL van de taak wekt uitzondering 13 op.
Wanneer een instructie of een interruptie betrekking heeft op een gate-descriptor, moet de DPL van de gate hetzelfde privilege hebben als (of minder dan) de CPL van de taak. Als DPL een hoger privilege-niveau heeft dan CPL treedt uitzondering 13 op. Als de bestemmings selector in de gate betrekking heeft op een code-segment descriptor, moet de DPL van de code-segment descriptor gelijk of meer bevoorrecht zijn dan de CPL van de taak. Zo niet, dan verschijnt uitzondering 13. Na de besturingsoverdracht is de DPL van de code-segment descriptor de nieuwe CPL van de taak. Als de bestemmings selector in de gate betrekking heeft op een task state segment, wordt automatisch een taakwisseling uitgevoerd.

De privilege-regels op een besturingsoverdracht vereisen:
- JMP of CALL direct naar een code-segment (code-segment descriptor) kan alleen naar een conformerend segment met DPL gelijk of meer geprivilegeerd dan CPL of naar een niet-conformerend segment op hetzelfde privilege-niveau.
- Interrupties binnen de taak of calls die privilege-niveaus zouden veranderen, kunnen alleen besturing overdragen via een gate op hetzelfde of een lager privilege-niveau dan CPL naar een code-segment op hetzelfde of een meer geprivilegeerd niveau dan CPL.
- Terugkeer-instructies die geen taken omschakelen kunnen alleen de besturing teruggeven aan een code-segment op hetzelfde of minder bevoorrecht niveau.
- Taakomschakeling kan worden uitgevoerd door een call, jump of interrupt die betrekking hebben op een task gate of een task gate segment met hetzelfde of een minder geprivilegeerd niveau.

3.3 80286

Veranderingen van privilege-niveau

Iedere besturingsoverdracht die CPL binnen de taak verandert, veroorzaakt een verandering van stacks als deel van de operatie. Initiële waarden van SS:SP voor privilege-niveaus 0, 1 en 2 blijven in het task state segment.
Tijdens een JMP of CALL besturingsoverdracht wordt de nieuwe stackpointer in de SS en SP registers geladen en wordt de vorige stackpointer op de nieuwe stack geduwd.
Bij terugkeer naar het originele privilege-niveau wordt de daarbij behorende stack als deel van de RET of IRET instructie weer teruggebracht. Voor subroutine-calls die parameters doorgeven aan de stack en privilege-niveaus doorkruisen, wordt een vast aantal woorden (gespecificeerd in de gate) gekopieerd van de vorige stack naar de huidige stack. De inter-segment RET instructie zal bij terugkeer de vorige stackpointer met een stack-adjustment waarde correct terugstellen.

Beveiliging

De 80286 heeft ook mechanismen die kritische instructies die de verwerkingsstatus van de CPU beïnvloeden (bijvoorbeeld HLT) en code of data-segmenten beveiligen tegen onjuist gebruik. Deze beveiligingsmechanismen zijn gegroepeerd in drie vormen:

- Beperkt ***gebruik*** van segmenten (bijvoorbeeld geen schrijven toegestaan op read only data-segmenten). De enige segmenten die gebruikt kunnen worden zijn gespecificeerd in de Local Descriptor Table (LDT) en de Global Descriptor Table (GDT).
- Beperkte ***toegang*** tot segmenten via de regels voor gebruik van privilege en descriptor.
- ***Geprivilegeerde instructies*** of operaties die alleen mogen worden uitgevoerd op bepaalde privilege-niveaus die gespecificeerd worden door CPL en I/O Privilege Level (IOPL). IOPL wordt bepaald door de bits 14 en 13 van het flag-woord.

Deze checks worden voor alle instructies uitgevoerd en kunnen in drie categoriën worden gesplitst:

- segment load checks (tabel 7/3.3-18);
- operand referentie-checks (tabel 7/3.3-19);
- geprivilegeerde instructie-checks (tabel 7/3.3-20).

Error Description	Exception Number
Descriptor table limit exceeded	13
Segment descriptor not-present	11 or 12
Privilege rules violated	13
Invalid descriptor/segment type segment register load: —Read only data segment load to SS —Special Control descriptor load to DS, ES, SS —Execute only segment load to DS, ES, SS —Data segment load to CS —Read/Execute code segment load to SS	13

Tabel 7/3.3-18: Segment-register load-checks.

Operand Reference Checks

Error Description	Exception Number
Write into code segment	13
Read from execute-only code segment	13
Write to read-only data segment	13
Segment limit exceeded[1]	12 or 13

NOTE:
Carry out in offset calculations is ignored.

Tabel 7/3.3-19: Operand referentie-checks.

Error Description	Exception Number
CPL ≠ 0 when executing the following instructions: LIDT, LLDT, LGDT, LTR, LMSW, CTS, HLT	13
CPL > IOPL when executing the following instructions: INS, IN, OUTS, OUT, STI, CLI, LOCK	13

Tabel 7/3.3-20: Geprivilegeerde instructie-checks.

3.3 80286

Interrupt Vector	Function	Return Address At Falling Instruction?	Always Restart-able?	Error Code on Stack?
8	Double exception detected	Yes	No[2]	Yes
9	Processor extension segment overrun	No	No[2]	No
10	Invalid task state segment	Yes	Yes	Yes
11	Segment not present	Yes	Yes	Yes
12	Stack segment overrun or stack segment not present	Yes	Yes[1]	Yes
13	General protection	Yes	No[2]	Yes

NOTE:
1. When a PUSHA or POPA instruction attempts to wrap around the stack segment, the machine state after the exception will not be restartable because stack segment wrap around is not permitted. This condition is identified by the value of the saved SP being either 0000(H), 0001(H), FFFE(H), or FFFF(H).
2. These exceptions indicate a violation to privilege rules or usage rules has occurred. Restart is generally not attempted under those conditions.

Tabel 7/3.3-21: Beschermde mode uitzonderingen.

Overtreding van de regels heeft een uitzondering tot gevolg. Een not-present uitzondering die betrekking heeft op de stack veroorzaakt uitzondering 12.

De IRET en POPF instructies voeren sommige van hun gedefinieerde functies niet uit als CPL onvoldoende privilege heeft.
Het zijn:
- De IF bit wordt niet veranderd als CPL > IOPL.
- Het IOPL-veld van het flag-woord wordt niet veranderd als CPL > 0.

Wanneer deze condities optreden worden geen uitzonderingen of andere indicaties gegeven.

Uitzonderingen

De 80286 detecteert verschillende typen uitzonderingen en interrupties in de beschermde mode (zie tabel 7/3.3-21). De meeste zijn herstartbaar nadat de uitzonderlijke toestand is weggenomen. Voor de meeste uitzonderingen kunnen interrupt handlers een errorcode lezen, die na het terugkeeradres op de stack is gezet en die de betrokken selector identificeert (0 indien geen enkele). Het terugkeeradres wijst gewoonlijk naar de falende instructie, inclusief alle voorafgaande prefixen. Bij een processor-uitbreidingssegment overrun-uitzondering wijst het terugkeeradres niet naar de ESC-instructie die de uitzondering veroorzaakte, maar kunnen de processor-extensie-registers het adres van de falende instructie bevatten.
Deze uitzonderingen signaleren dat een overtreding van privilege- of gebruiksregels is opgetreden. Onder die omstandigheden wordt meestal niet geprobeerd opnieuw te starten.

Speciale operaties

Taak-omschakeling

De 80286 heeft een ingebouwde operatie voor taakomschakelingen die de gehele 80286 executie-status veilig stelt (registers, adresruimte en een verbinding met de vorige taak), een nieuwe executie-status laadt en begint met de uitvoering van de nieuwe taak. Net als bij gates wordt de taakomschakelings-operatie betrokken bij de uitvoering van een inter-segment JMP of CALL instructie die betrekking heeft op een Task State Segment (TSS) of task gate descriptor in de GDT of LDT. Een INT n instructie, uitzondering of externe interruptie kan ook de taakomschakeling oproepen door een task gate descriptor in de betrokken IDT descriptor-ingang te selecteren.
De TSS descriptor wijst naar een segment dat de gehele 80286 executie-status bevat (zie figuur 7/3.3-21), terwijl een task gate

descriptor een TSS selector bevat. Het limietveld van de descriptor moet groter dan 002B(H) zijn.

Elke taak moet een eigen TSS hebben. De huidige TSS wordt geïdentificeerd door een speciaal register in de 80286: het Task Register (TR). Dit register bevat een selector die betrekking heeft op de task gate segment-descriptor die de huidige TSS bepaalt. Verborgen basis- en limietregisters die verbonden zijn met TR worden telkens geladen als TR met een nieuwe selector wordt geladen.
De IRET instructie wordt gebruikt om besturing terug te geven aan de taak die de huidige taak opriep of werd geïnterrumpeerd. Bit 14 in het flag-register wordt de geneste taakbit (NT) genoemd. Hij regelt de functie van de IRET-instructie. Als NT = 0, voert de IRET-instructie de gewone huidige taak uit door waarden van de stack te poppen.
Als NT = 1, voert IRET een taakomschakeling terug naar de vorige taak uit.
Wanneer een taakomschakeling wordt ingeleid door een CALL, JMP of INT instructie, worden de oude (behalve bij JMP) en nieuwe TSS "busy" gemarkeerd en wordt het terugkoppelveld van de nieuwe TSS op de oude TSS selector gezet. De NT-bit van de nieuwe taak wordt geset door via CALL of INT geïnitialiseerde taakomschakelingen. Een interruptie die geen taakomschakeling veroorzaakt, zal NT leegmaken. NT kan ook worden geset of gecleared door POPF of IRET instructies.
Het task state segment wordt "busy" gemarkeerd door het descriptor-veld te veranderen van Type 1 in Type 3. Wordt een selector gebruikt die betrokken is bij een busy state segment dan verschijnt uitzondering 13.

Processor-uitbreiding context omschakeling

De context van een processor-uitbreiding (zoals de 80287 numerieke coprocessor) wordt niet veranderd door de taakomschakel-operatie.

Een processor-uitbreidings context hoeft alleen te worden veranderd als een andere taak de processor-uitbreiding probeert te gebruiken (die nog steeds de context van de vorige taak bevat). De 80286 detecteert het eerste gebruik van een processor-uitbreiding na een taak-omschakeling door het opwekken van de processor-extensie not-present uitzondering (7). De interrupt handler kan dan beslissen of een context-verandering nodig is.
Altijd wanneer de 80286 taken omschakelt wordt de Task Switched (TS) bit van het MSW geset. TS geeft aan dat een processor-uitbreiding context aan een andere dan de huidige taak kan toebehoren.
De processor-uitbreiding not-present uitzondering (7) treedt op wanneer wordt geprobeerd een ESC of WAIT instructie uit te voeren als TS = 1 en een processor-uitbreiding aanwezig is (MP = 1 in het MSW).

Instruction	Operands	Function
ARPL	Selector, Register	Adjust Requested Privilege Level: adjusts the RPL of the selector to the numeric maximum of current selector RPL value and the RPL value in the register. Set zero flag if selector RPL was changed by ARPL.
VERR	Selector	VERify for Read: sets the zero flag if the segment referred to by the selector can be read.
VERW	Selector	VERify for Write: sets the zero flag if the segment referred to by the selector can be written.
LSL	Register, Selector	Load Segment Limit: reads the segment limit into the register if privilege rules and descriptor type allow. Set zero flag if successful.
LAR	Register, Selector	Load Access Rights: reads the descriptor access rights byte into the register if privilege rules allow. Set zero flag if successful.

Tabel 7/3.3-22: Pointer test-instructies van de 80286.

3.3 80286

Instructies voor Pointer testing
De 80286 heeft verschillende instructies voor het snel uitvoeren van pointer testing en overeenstemmingscontrole (zie tabel 7/3.3-22). Deze instructies gebruiken de geheugen-management hardware om te verifiëren dat een selector-waarde refereert aan een geschikt segment zonder een uitzondering te riskeren. Een conditie-flag (ZF) geeft aan of gebruik van de selector of het segment een uitzondering zal veroorzaken.

Dubbele fout en shutdown
Als tijdens de uitvoering van een enkele instructie twee aparte uitzondering worden gedetecteerd, levert de 80286 de dubbele fout uitzondering (8). Vindt een executie plaats tijdens de bewerking van een dubbele fout uitzondering, dan komt de 80286 in shutdown. Gedurende shutdown worden geen verdere instructies of uitzonderingen bewerkt. De 80286 kan alleen door NMI (de CPU blijft in de beschermde mode) of RESET (de CPU verlaat de beschermde mode) uit shutdown worden gehaald. Shutdown wordt naar buiten gesignaleerd via de HALT bus-operatie met A1 LAAG.

Initialisatie van de beschermde mode
Na RESET voert de 80286 eerst instructies uit in de reële mode. Om te bereiken dat initialisatiecode aan de top van het fysieke geheugen wordt geplaatst, worden A23 tot en met A20 HOOG wanneer de 80286 geheugen-verwijzingen ten opzichte van het CS-register uitvoert totdat CS wordt veranderd. A23 tot en met A20 zijn LAAG voor verwijzingen naar de DS, ES of SS segmenten. Door CS in de reële adresseringsmode te veranderen worden A23 tot en met A20 steeds LAAG wanneer CS opnieuw wordt gebruikt. De initiële CS:IP waarde van F000:FFF0 levert 64 kB ruimte om code te initialiseren zonder CS te veranderen.
Voor de beschermde mode moeten verschillende registers worden geïnitialiseerd. De GDT en IDT basisregisters moeten betrekking hebben op een geldige GDT en IDT. Na uitvoering van de LMSW instructie om PE te setten moet de 80286 onmiddellijk een intra-segment JMP instructie uitvoeren om de instructie-wachtrij vrij te maken van instructies die in de reële adresseringsmode waren geadresseerd.
Om de registers van de 80286 CPU te dwingen overeen te komen met de door de software aangenomen initiële beschermde mode status, wordt een JMP instructie uitgevoerd met een selector die betrekking heeft op de initiële TSS die in het systeem wordt gebruikt.
Hierdoor worden het task register, het lokale descriptor tabel-register, de segment-registers en de initiële algemene registerstatus geladen. Het TR dient naar een geldig TSS te wijzen, omdat een taakomschakelings-operatie gepaard gaat met het opbergen van de huidige task state.

Systeem interface

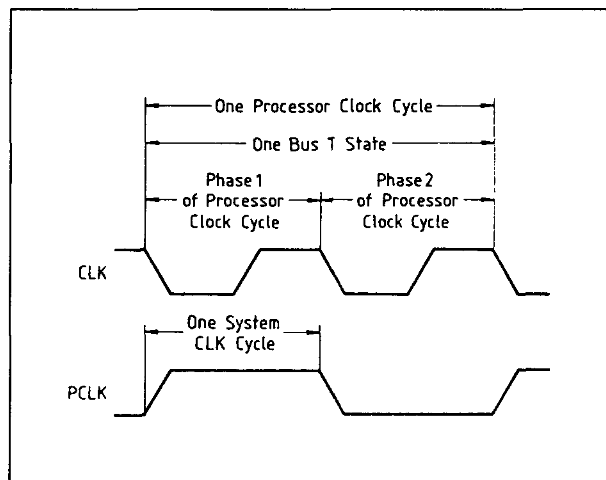
Inleiding
De systeem-interface van de 80286 kan twee vormen hebben: een lokale bus en een systeembus. De lokale bus bestaat uit adres, data- status- en besturingssignalen op de aansluitpennen van de processor. Een gebufferde vorm van de lokale bus wordt systeembus genoemd. Een systeembus kan ook van de lokale bus afwijken in termen van codering van status- en besturingslijnen en/of timing en laden van signalen. De 80286-familie bevat verschillende componenten voor het opzetten van standaard systeembussen, zoals de IEEE 796 standaard Multibus.

Bus interface-signalen en timing
De 80286 microsystem lokale bus koppelt de 80286 aan lokale geheugen- en I/O-componenten. De interface heeft 24 adreslijnen, 16 datalijnen en 8 status- en besturingssignalen.
De 80286 CPU, 82284 clock generator, 82288 buscontroller, 82289 busarbiter, tran-

sceivers en latches leveren een gebufferde en gedecodeerde systeembus interface.
De 82284 wekt de systeemclock op en synchroniseert READY en RESET. De 82288 converteert door de 80286 gecodeerde busoperatiestatus naar commando- en busbesturingssignalen. De 82289 bus-arbiter genereert Multibus arbitrage-signalen. Deze componenten kunnen de timing en elektrische niveaus verzorgen die voor de meeste systeembus interfaces nodig zijn.

Fysiek geheugen en I/O interface
In de beschermde mode kunnen maximaal 16 MB aan fysiek geheugen worden geadresseerd. In de reële mode kan 1 MB worden geadresseerd. Het geheugen is toegankelijk als bytes of als woorden. Woorden bestaan uit twee opeenvolgende bytes die door de minst belangrijke byte die in het laagste adres zijn opgeslagen, worden geadresseerd.
Byte-transfers vinden aan beide helften van de lokale 16-bit databus plaats. Even bytes worden bereikt via D7 tot en met D0, terwijl oneven bytes via D15 tot en met D8 worden overgebracht. Even-geadresseerde woorden worden in één buscyclus via D15 tot en met D0 overgebracht, terwijl voor oneven geadresseerde woorden twee buscycli nodig zijn. De eerste brengt data over op D15 tot en met D8 en de tweede op D7 tot en met D0. Beide byte data-transfers vinden automatisch, transparant voor de software, plaats.
De bussignalen A0 en BHE regelen de overdrachten via de lage en hoge helften van de databus. Even adresbyte-transfers worden aangeduid met A0 LAAG en BHE HOOG, oneven adresbyte-transfers met A0 HOOG en BHE LAAG. Zowel A0 als BHE zijn LAAG bij even adreswoord-transfers.
De I/O adresruimte omvat 64 kB adressen in beide modes. De I/O ruimte en het geheugen zijn toegankelijk voor bytes en woorden. "Byte-wide" periferie-schakelingen mogen zowel met de hoge als met de lage byte van de databus worden verbonden. Byte-wide I/O schakelingen die aan de hoge databyte (D15 tot en met D8) vast zitten worden bereikt met oneven I/O adressen, I/O schakelingen aan de lage databyte met even I/O adressen. Een interrupt controller zoals de 8259A moet op de lage databyte (D7 tot en met D0) zijn aangesloten voor correcte terugkeer van de interrupt-vector.

Bus-operatie
De 80286 maakt gebruik van een dubbele systeemclock (CLK-ingang) voor timing van de bus. Alle signalen op de lokale bus worden ten opzichte van de systeem CLK-ingang gemeten. De CPU deelt de systeemclock door twee om de interne processorclock te produceren waarmee de busstatus wordt bepaald.
Elke processorclock is samengesteld uit twee systeemclock-cycli: fase 1 en fase 2. Het uitgangssignaal (PCLK) van de 82284 clockgenerator identificeert de volgende fase van de processorclock (zie figuur 7/3.3-22).



Figuur 7/3.3-22: De samenhang tussen systeem- en processorclock.

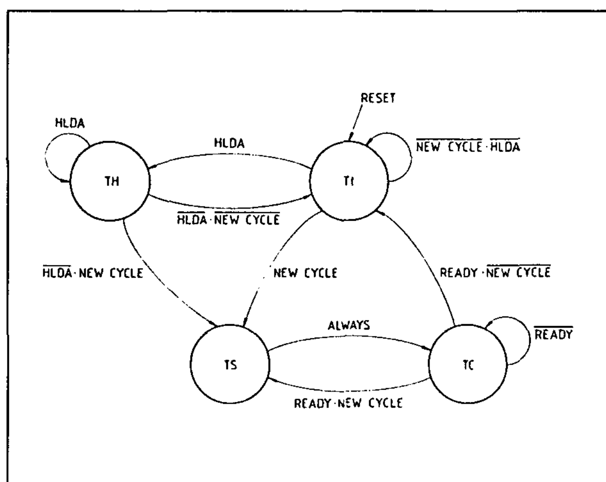
Er worden zes typen bus-operaties ondersteund:
- geheugen uitlezen (memory read);
- schrijven in geheugen (memory write);
- in-/uitgang uitlezen (I/O read);
- schrijven naar in-/uitgang (I/O write);

- bevestig interruptie (interrupt acknowledge);
- halt/shutdown.

Data kan worden getransporteerd met een maximum snelheid van één woord per twee processor-clockcycli.

De 80286 bus heeft drie basistoestanden:
- leegloop (idle: T_i);
- verzend status (send status: T_s);
- voer commando uit (perform command: T_c).

De 80286 CPU heeft nog een vierde lokale bus-toestand: hold (T_h). T_h geeft aan dat de 80286, als antwoord op een HOLD request, de besturing van de lokale bus heeft overgegeven aan een andere busmaster.
Elke busstatus duurt één processorclock. Figuur 7/3.3-23 laat de vier 80286 lokale bustoestanden en toegestane overgangen zien.



Figuur 7/3.3-23: Bustoestanden van de 80286.

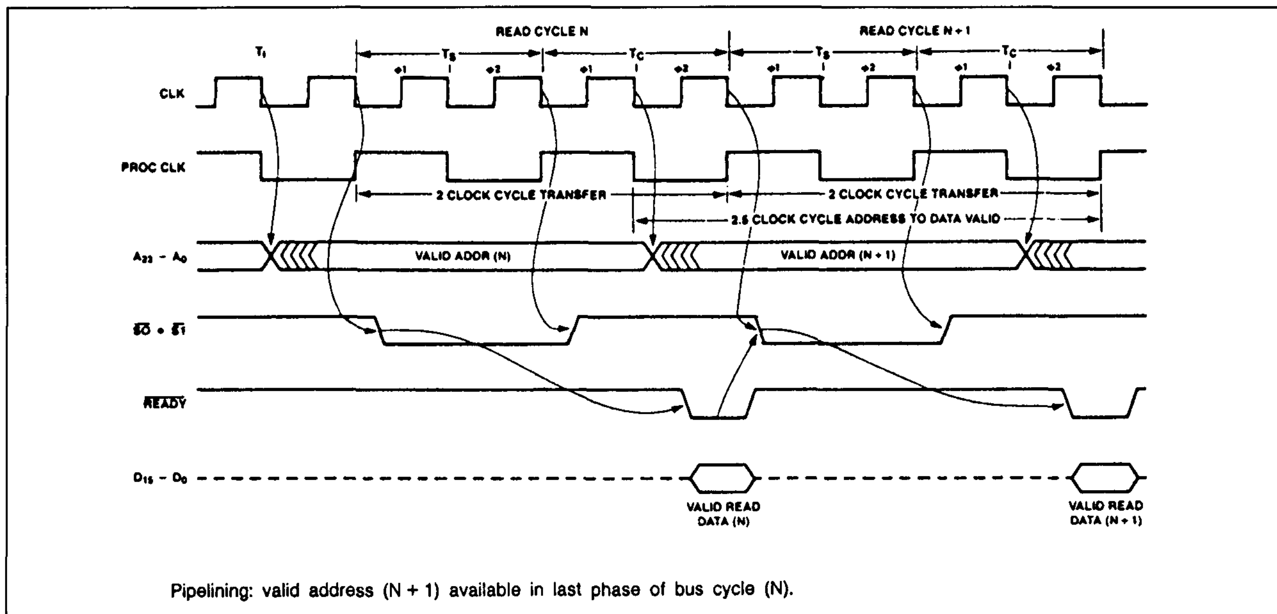
Bustoestanden

De leegloop-toestand (T_i) geeft een indicatie dat geen data-overdrachten bezig zijn of zijn aangevraagd. De eerste aktieve toestand T_s wordt gesignaleerd door het LAAG gaan van statuslijn $\overline{S1}$ of $\overline{S0}$ en het identificeren van fase 1 van de processorclock. Gedurende T_s zijn de commando-codering, het adres en data (voor een schrijfcyclus) beschikbaar aan de uitgangspennen van de 80286. De 82288 buscontroller decodeert de statussignalen en genereert een Multibus compatibel lees/schrijf commando en lokale transceiver besturingssignalen.
Na T_s komt de 80286 in de uitvoeringstoestand (T_c).
Tijdens T_c reageren geheugen of I/O-schakelingen op de busoperatie door leesdata naar de CPU te sturen of schrijfdata te accepteren.
T_c toestanden mogen zo vaak worden herhaald als nodig is om er zeker van te zijn dat het geheugen of de I/O-schakeling heeft gereageerd. Het $\overline{READY}$ signaal bepaalt of T_c wordt herhaald. Een herhaalde T_c-toestand wordt een wachttoestand (wait state) genoemd.
Gedurende hold (T_h) laat de 80286 alle adres-, data- en status-uitgangspennen zweven om een andere busmaster in staat te stellen gebruik te maken van de lokale bus. Het 80286 HOLD ingangssignaal wordt gebruikt om de 80286 in de T_h-toestand te brengen.
Het 80286 HLDA uitgangssignaal geeft aan dat de CPU in T_h is gekomen.

Pijplijn adressering

De 80286 gebruikt een lokale bus-interface met gepijplijnde timing om zoveel mogelijk tijd beschikbaar te hebben voor datatoegang.
Door gepijplijnde timing kan elke twee processorcycli een nieuwe busoperatie worden geïnitialiseerd, terwijl elke individuele busoperatie drie processorcycli kan duren.
De timing van de adres-uitgangen is zodanig gepijplijnd dat het adres van de volgende busoperatie al tijdens de lopende busoperatie beschikbaar komt. Of, met andere woorden: de eerste clock van de volgende busoperatie wordt overlapt door de laatste clock van de huidige busoperatie.
De adres-decodering en routing logika kunnen dus vooruitlopend op de volgende busoperatie werken.

3.3 80286



Pipelining: valid address (N + 1) available in last phase of bus cycle (N).

Figuur 7/3.3-24: Basis buscyclus van de 80286.

Het adres kan gedurende de gehele busoperatie stabiel worden gehouden door externe adreslatches die bovendien voor extra AC en DC buffering zorgen. De 80286 houdt het adres van de lopende busoperatie niet vast tijdens alle T_C-toestanden. Het adres van de volgende busoperatie kan gedurende fase 2 van elke willekeurige T_C worden verzonden. Het adres blijft geldig gedurende fase 1 van de eerste T_C om houdtijd ten opzichte van ALE voor de adreslatch-ingangen te garanderen.

Bus-besturingssignalen

De 82288 buscontroller levert de besturingssignalen:

- address latch enable (ALE);
- lees/schrijf opdrachten;
- data transmit/receive (DT/$\overline{R}$);
- data enable (DEN).

De Address Latch Enable (ALE) uitgang bepaalt wanneer het adres gelatcht mag worden. ALE levert minstens één systeem CLK-periode adres-houdtijd vanaf het einde van de vorige busoperatie totdat het adres voor de volgende busoperatie op de latch-uitgangen verschijnt.

Deze adres-houdtijd is nodig voor Multibus- en gewone geheugensystemen.

De databus transceivers worden bestuurd door de Data Enable (DEN) en Data Transmit/Receive (DT/$\overline{R}$ uitgangen) van de 82288. DEN geeft de data transceivers vrij, terwijl de richting van de transceivers wordt bepaald door DT/$\overline{R}$.

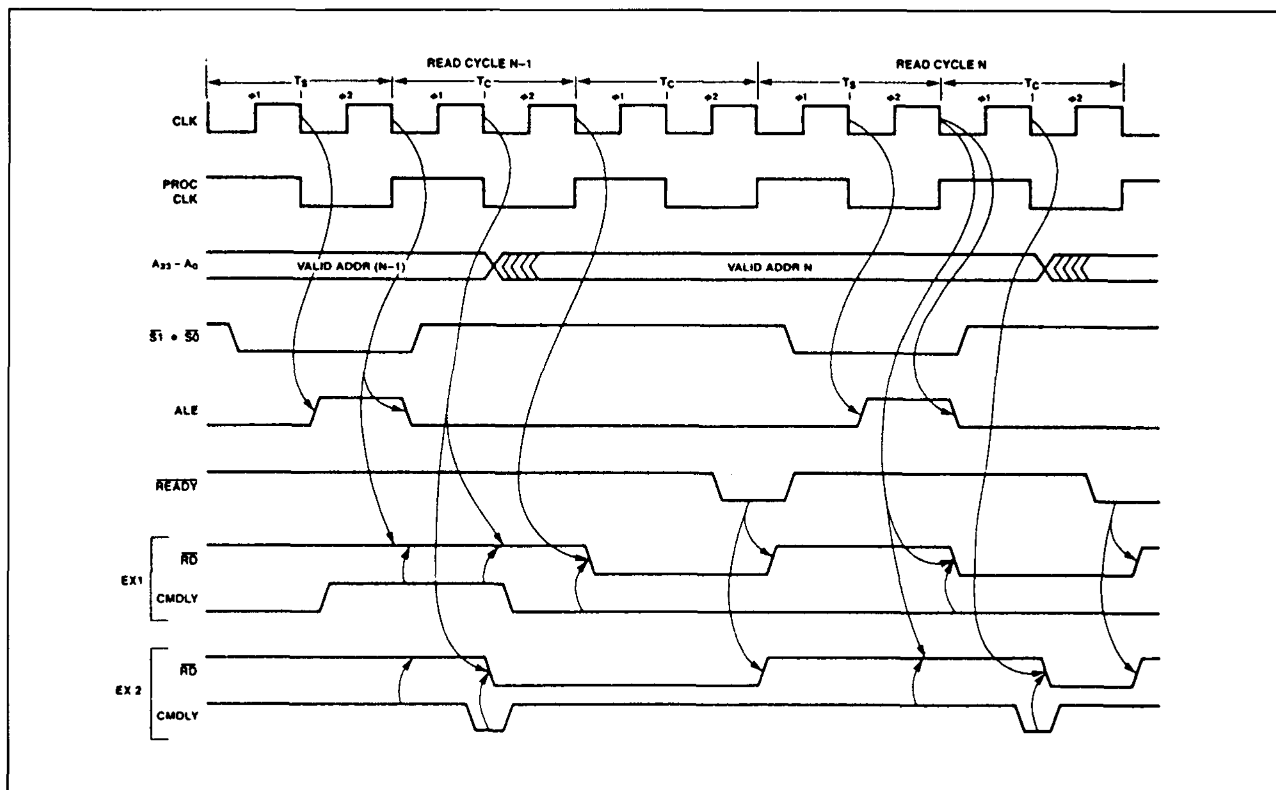
Op DEN en DT/$\overline{R}$ vindt timing plaats om te voorkomen dat bus-rivaliteit optreedt tussen de busmaster, databus-transceivers en systeem databus-transceivers.

Timing van commando's

Op de lokale bus van de 80286 zijn voor aanpassingen twee timing-opties beschikbaar: verlenging van commando's (command extension) en vertraging van commando's (command delay).

Command extension geeft externe schakelingen meer tijd om op een commando te reageren en komt overeen met het invoegen van wachttijden bij de 8086. De tijdsduur van iedere busoperatie kan door externe logika zodanig worden geregeld dat de operatie zo kort mogelijk duurt.

3.3 80286



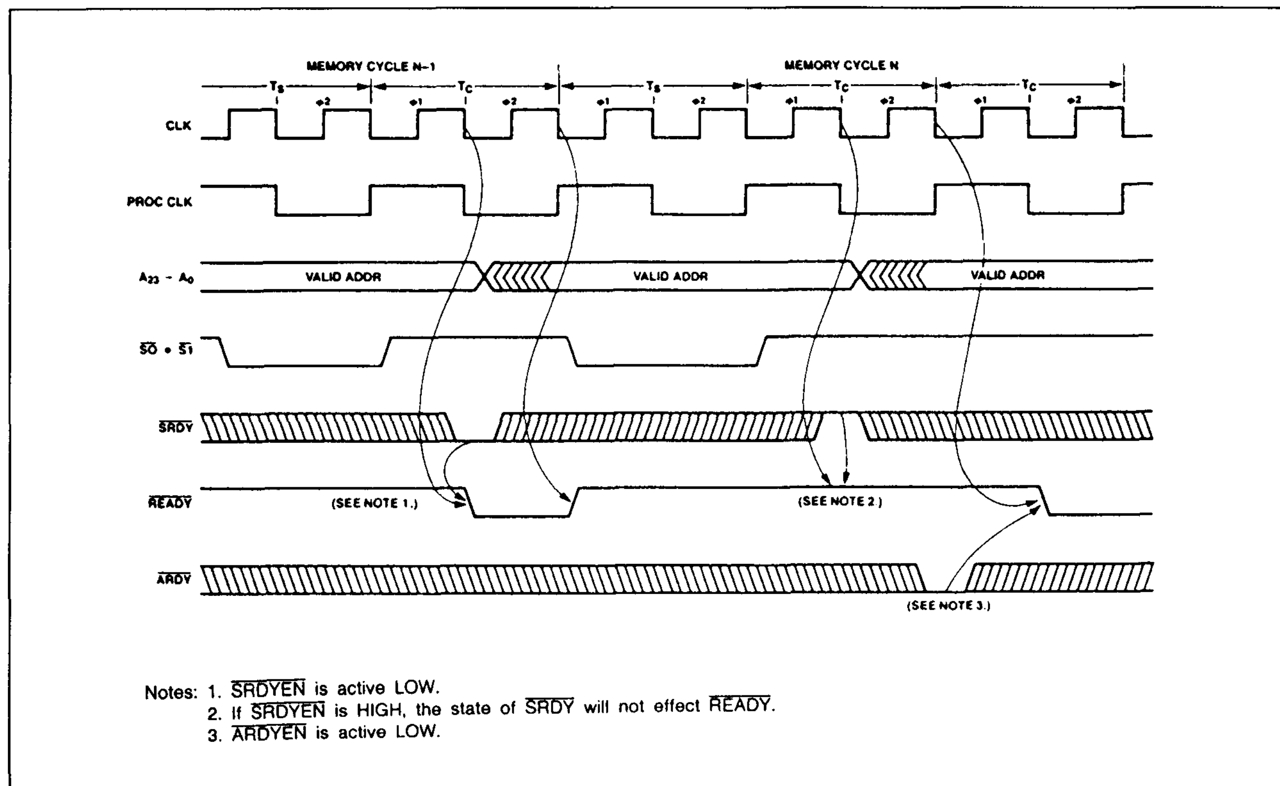
Figuur 7/3.3-25: Besturing van de voorflank van commando-signalen door het CMDLY-signaal.

Het $\overline{\text{READY}}$ ingangssignaal kan iedere busoperatie zo lang als nodig is verlengen.
Command delay levert door vertraging van het aktief worden van systeembus-commando's meer adres- of data setup-tijd aan operaties op de systeembus. Deze vertraging wordt geregeld door de CMDLY ingang van de 82288. Na afloop van T_S kijkt de buscontroller bij elke dalende flank van CLK naar CMDLY. Als CMDLY dan HOOG is zal de 82288 het commando-signaal niet aktiveren; is CMDLY LAAG dan wordt het commando-signaal wel geaktiveerd. Zodra het commando aktief is wordt de CMDLY ingang niet meer bemonsterd.
Wanneer een commando wordt vertraagd is de beschikbare responsietijd vanaf het aktief worden van het commando tot het inlezen of wegschrijven van data korter. Om de systeembus-timing aan te passen kan een adres-decoder bepalen voor welke busoperaties vertraging van de commando's nodig is. De CMDLY ingang heeft geen invloed op de timing van ALE, DEN of DT/$\overline{\text{R}}$. In figuur 7/3.3-25 worden vier toepassingen van CMDLY getoond. In voorbeeld 1 (EX1) wordt het leescommando twee systeem CLK's vertraagd voor cyclus N-1 en is er geen vertraging voor cyclus N. Voorbeeld 2 (EX2) toont vertraging van het leescommando met één systeem CLK voor cyclus N-1 en één systeem CLK voor cyclus N.

Beëindiging van de buscyclus

Bij maximale overdrachtsnelheden verkeert de 80286 bus afwisselend in status- en commando toestanden. De bus-statussignalen worden niet-aktief na T_S zodat hiermee, na afloop van de huidige cyclus, het begin van de volgende busoperatie correct kan worden gesignaleerd.
Er is geen externe indicatie van T_C op de lokale bus van de 80286. De busmaster en de buscontroller komen direct na T_S in T_C en gaan door met het uitvoeren van T_C-cycli totdat $\overline{\text{READY}}$ verschijnt.

3.3 80286



Figuur 7/3.3-26: Synchrone- en Asynchrone Ready.

$\overline{\text{READY}}$ operatie

De huidige busmaster en de 82288 buscontroller beëindigen elke busoperatie gelijktijdig om een maximale busoperatie-bandbreedte te verkrijgen. Beide worden van te voren geïnformeerd door het aktief worden van $\overline{\text{READY}}$ (open-collector uitgang van 82284) die de laatste T_C-cyclus van de lopende busoperatie identificeert. De busmaster en de buscontroller moeten dezelfde betekenis van het $\overline{\text{READY}}$ signaal zien, waarbij vereist wordt dat $\overline{\text{READY}}$ synchroon is met de systeemclock.

Synchrone Ready

De 82284 clock-generator synchroniseert $\overline{\text{READY}}$ op zowel synchrone bronnen als asynchrone (zie figuur 7/3.3-26). De synchrone ready-ingang ($\overline{\text{SRDY}}$) van de clock-generator wordt, aan het einde van fase 1 van elke T_C, bemonsterd op de dalende flank van CLK. De status van $\overline{\text{SRDY}}$ wordt dan via de $\overline{\text{READY}}$ uitgangslijn uitgezonden naar de busmaster en de buscontroller.

Asynchrone Ready

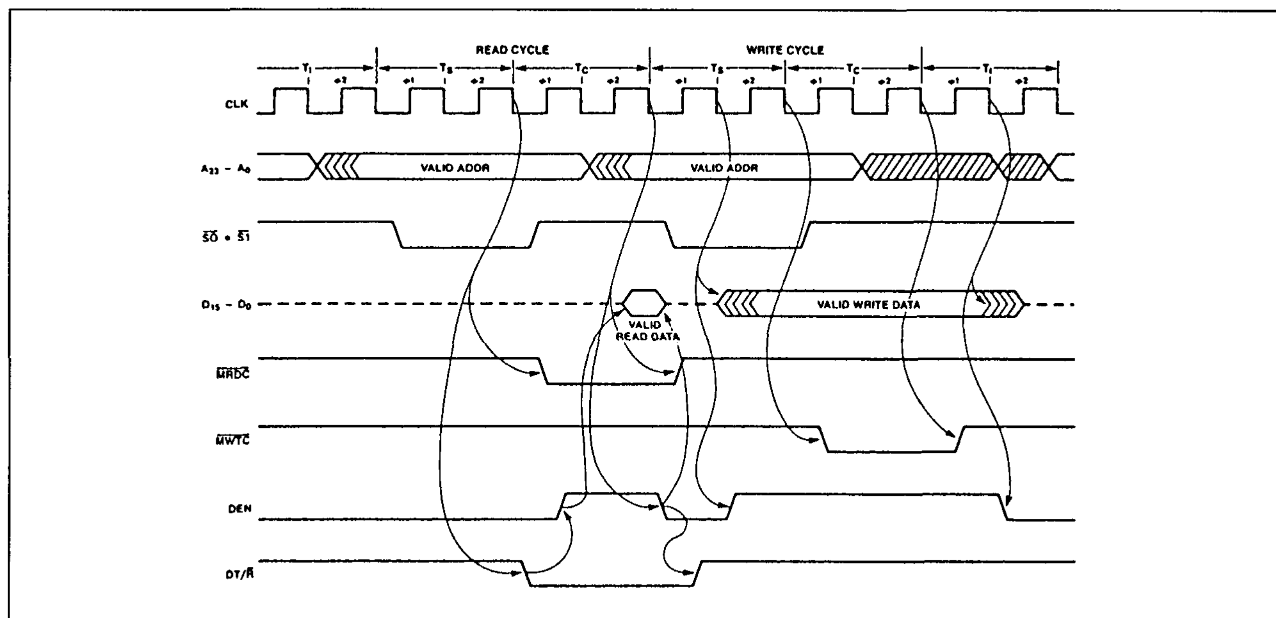
Veel systemen hebben schakelingen of subsystemen die asynchroon werken ten opzichte van de systeemclock.

Als gevolg daarvan kan van hun ready-uitgangen niet worden gegarandeerd dat ze voldoen aan de setup- en houdtijd-eisen voor $\overline{\text{SRDY}}$ van de 82284.

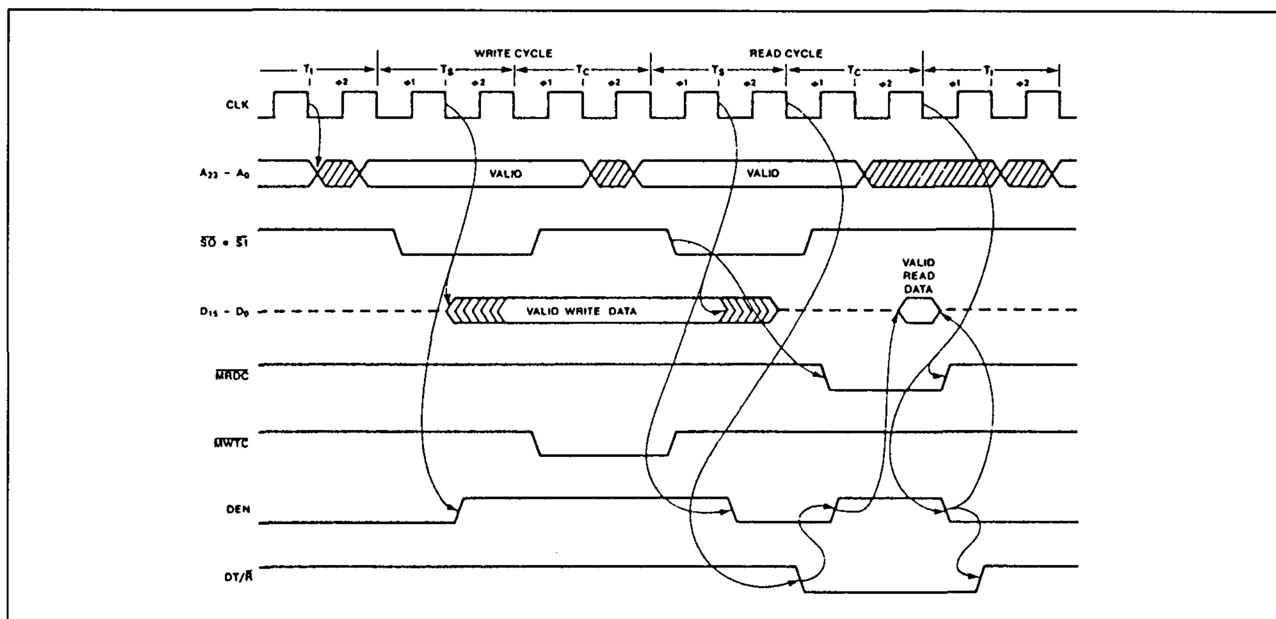
De asynchrone ready ingang ($\overline{\text{ARDY}}$) van de 82284 is echter speciaal ontworpen voor dergelijke signalen. De $\overline{\text{ARDY}}$-ingang wordt aan het begin van elke T_C-cyclus door de synchronisatie-logika van de 82284 bemonsterd.

Hierdoor wordt binnen één systeem CLK cyclus de waarde ervan bepaald voordat die naar de busmaster en de buscontroller wordt gezonden. $\overline{\text{ARDY}}$ of $\overline{\text{ARDYEN}}$ moeten HOOG zijn aan het einde van T_S.

3.3 80286



Figuur 7/3.3-27: Rug aan rug Lees-Schrijf cycli.



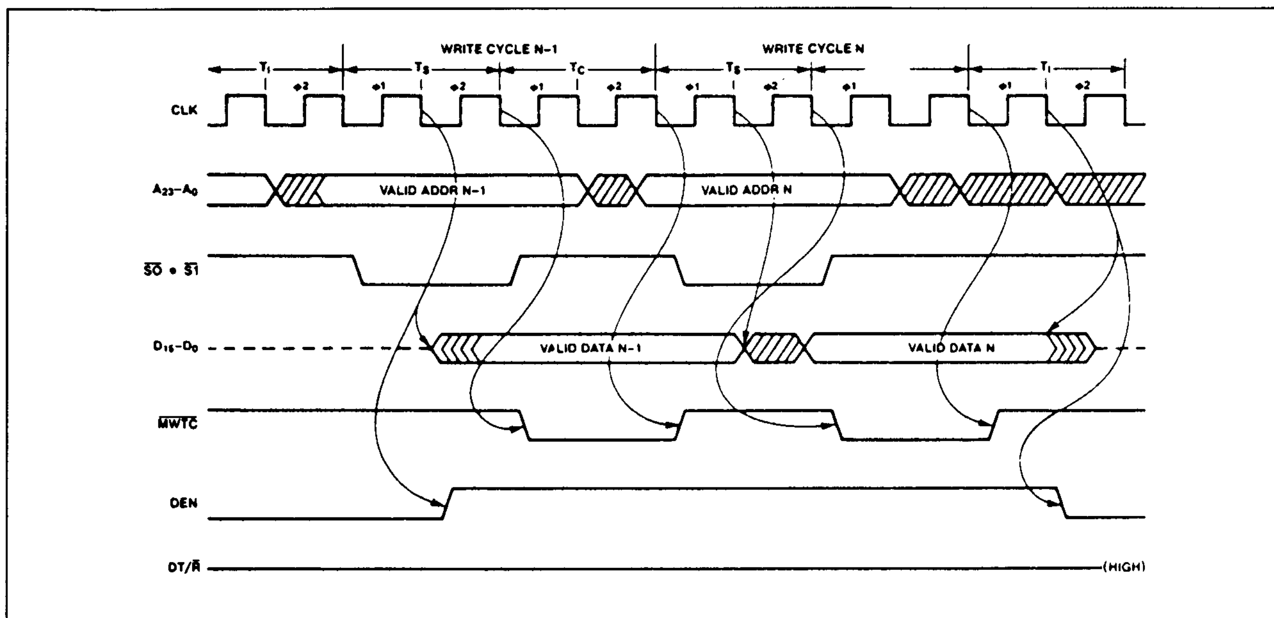
Figuur 7/3.3-28: Rug aan rug Schrijf-Lees cycli.

ARDY kan niet worden gebruikt voor het beëindigen van buscycli zonder wachttoestanden.

Elke ready-ingang van de 82284 heeft een eigen enable-pen ($\overline{SRDYEN}$ en $\overline{ARDYEN}$) om te selecteren of de lopende busoperatie door de synchrone of door de asynchrone ready zal worden beëindigd. Een busoperatie kan door elk van de ready-signalen worden beëindigd.

Deze enable-ingangen zijn aktief-LAAG en hebben dezelfde timing als hun respectievelijke ready-ingangen. Meestal wordt door adres-decodeer logika geselecteerd of de lopende busoperatie door $\overline{ARDY}$ of $\overline{SRDY}$ moet worden beëindigd.

3.3 80286



Figuur 7/3.3-29: Rug aan rug Schrijf-Schrijf cycli.

Databus-besturing

De figuren 7/3.3-27, -28 en -29 laten zien hoe de DT/R̄, DEN, databus en adres signalen werken bij verschillende combinaties van lezen, schrijven en leegloop busoperaties. DT/R̄ gaat aktief (LAAG) bij een leesoperatie. DT/R̄ blijft HOOG vóór, tijdens en tussen schrijfoperaties.

De databus wordt tijdens de tweede fase van T_S aangestuurd met schrijfdata. Door de vertraging van de schrijfdata timing hebben de leesdata drivers, vanaf een vorige leescyclus, voldoende tijd om in de 3-state OFF toestand te komen voordat de 80286 CPU begint met het aansturen van de lokale bus voor schrijfoperaties. Schrijfdata zal altijd gedurende één systeemclock na de laatste T_C geldig blijven om voldoende houdtijd voor Multibus of andere gelijksoortige geheugen- of I/O-systemen te leveren. Tijdens schrijf-lees of schrijf-leegloop volgorden komt de databus in de 3-state OFF positie gedurende de tweede fase van de processor-cyclus na de laatste T_C. In een schrijf-schrijf volgorde komt de databus tussen T_C en T_S niet in de 3-state OFF toestand.

Gebruik van de bus

Inleiding

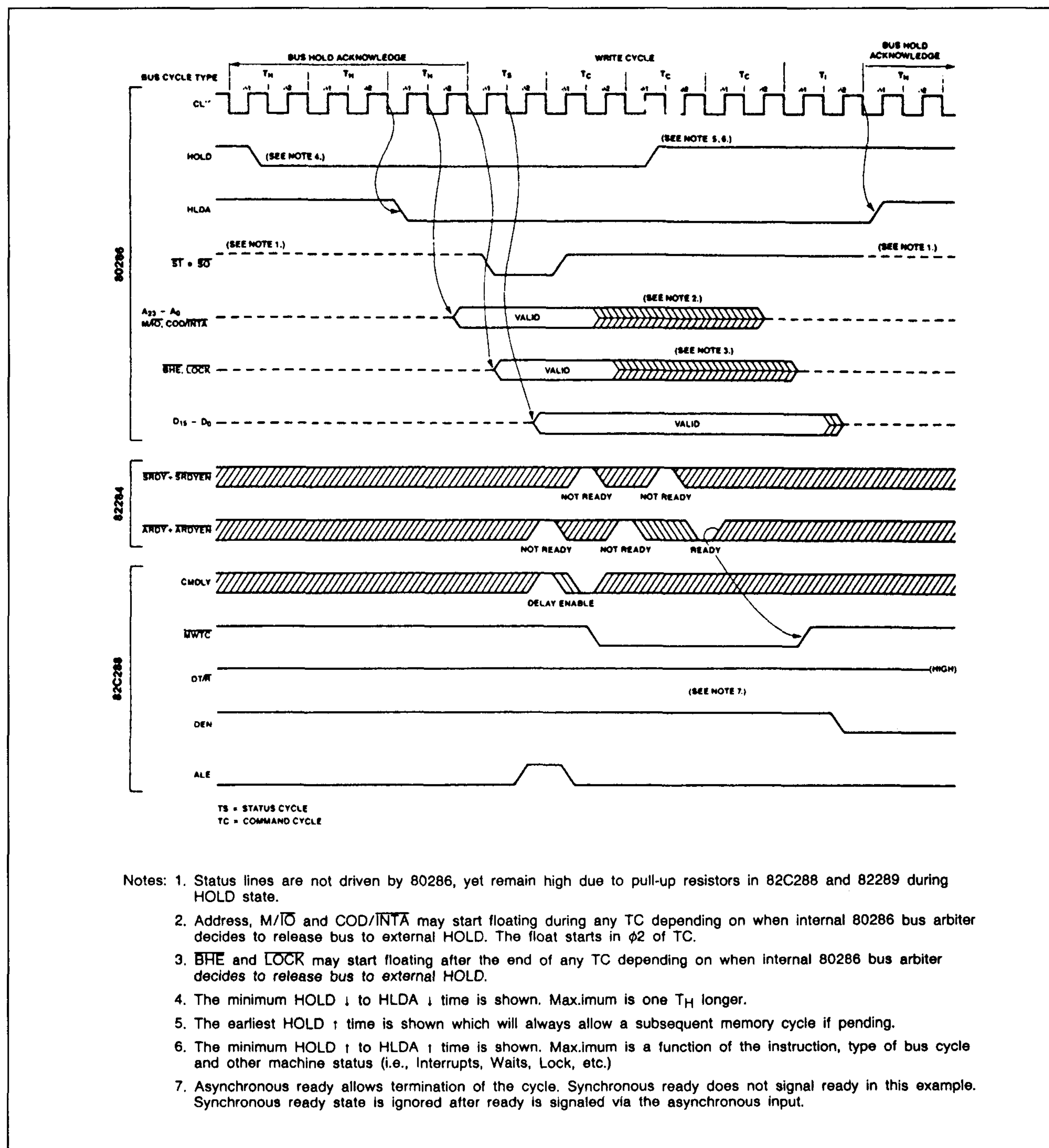
De lokale bus van de 80286 kan voor verschillende functies worden gebruikt:
- instructie data-transfers;
- data-transfers door andere busmasters;
- instructie fetching;
- processor-extensie data-transfers;
- interrupt acknowledge;
- halt/shutdown.

In dit gedeelte worden lokale busaktiviteiten met speciale signalen besproken of waaraan speciale eisen worden gesteld.

HOLD en HLDA

Door HOLD en HLDA wordt een andere busmaster in staat gesteld de besturing van de lokale bus over te nemen door de 80286 bus in de T_h-toestand te brengen. In figuur 7/3.3-30 is de volgorde te zien van de gebeurtenissen die nodig zijn om de besturing over te dragen. In dit voorbeeld is de 80286 in het begin in de T_h-toestand hetgeen wordt gesignaleerd door het aktief zijn van HLDA. Bij het verlaten van T_h (HLDA wordt niet-aktief) wordt een schrijf-operatie gestart.

3.3 80286



Figuur 7/3.3-30: Beëindiging van Multibus schrijven door Asynchrone Ready met Bus Hold (overdracht van de lokale bus aan een andere busmaster).

Tijdens het schrijven wordt de 80286 door een andere busmaster om de lokale bus gevraagd, hetgeen te zien is aan het HOLD-signaal. Na het volbrengen van de schrijfoperatie voert de 80286 één T buscyclus uit om schrijfdata-houdtijd te garanderen en komt dan in T_h terecht (gesignaleerd door het aktief worden van HLDA).

3.3 80286

CMDLY en $\overline{\text{ARDY}}$ ready worden gebruikt voor het starten en stoppen van het schrijfbus commando. Let op dat $\overline{\text{SRDY}}$ niet-aktief of door $\overline{\text{SRDYEN}}$ gesperd moet zijn om te garanderen dat met $\overline{\text{ARDY}}$ de cyclus wordt beëindigd. HOLD mag niet aktief zijn gedurende de tijd vanaf de voorflank van RESET tot 34 CLK's na de achterflank van RESET.

Lock

De CPU geeft een aktief blokkeer (lock)-signaal tijdens Interrupt-Acknowledge cycli, de XCHG instructie en sommige descriptortoegangen.
Lock wordt ook gegeven wanneer de LOCK prefix wordt gebruikt. De LOCK prefix kan bij de volgende ASM-286 assembler-instructies worden gebruikt: MOVS, INS en OUTS. Bij andere buscycli dan Interrupt-Acknowledge cycli zal Lock bij de eerste en volgende cycli van een te blokkeren reeks aktief zijn.
Lock blijkt niet aktief te zijn gedurende de laatste te blokkeren cyclus. Bij de voorlaatste cyclus wordt Lock niet-aktief op het einde van de eerste T_C, ongeacht het aantal ingelaste wachttoestanden.
Bij Interrupt-Acknowledge cycli wordt Lock bij elke cyclus aktief en op het einde van de eerste T_C van elke cyclus niet-aktief, ongeacht het aantal ingelaste wachttoestanden.

Instruction Fetching

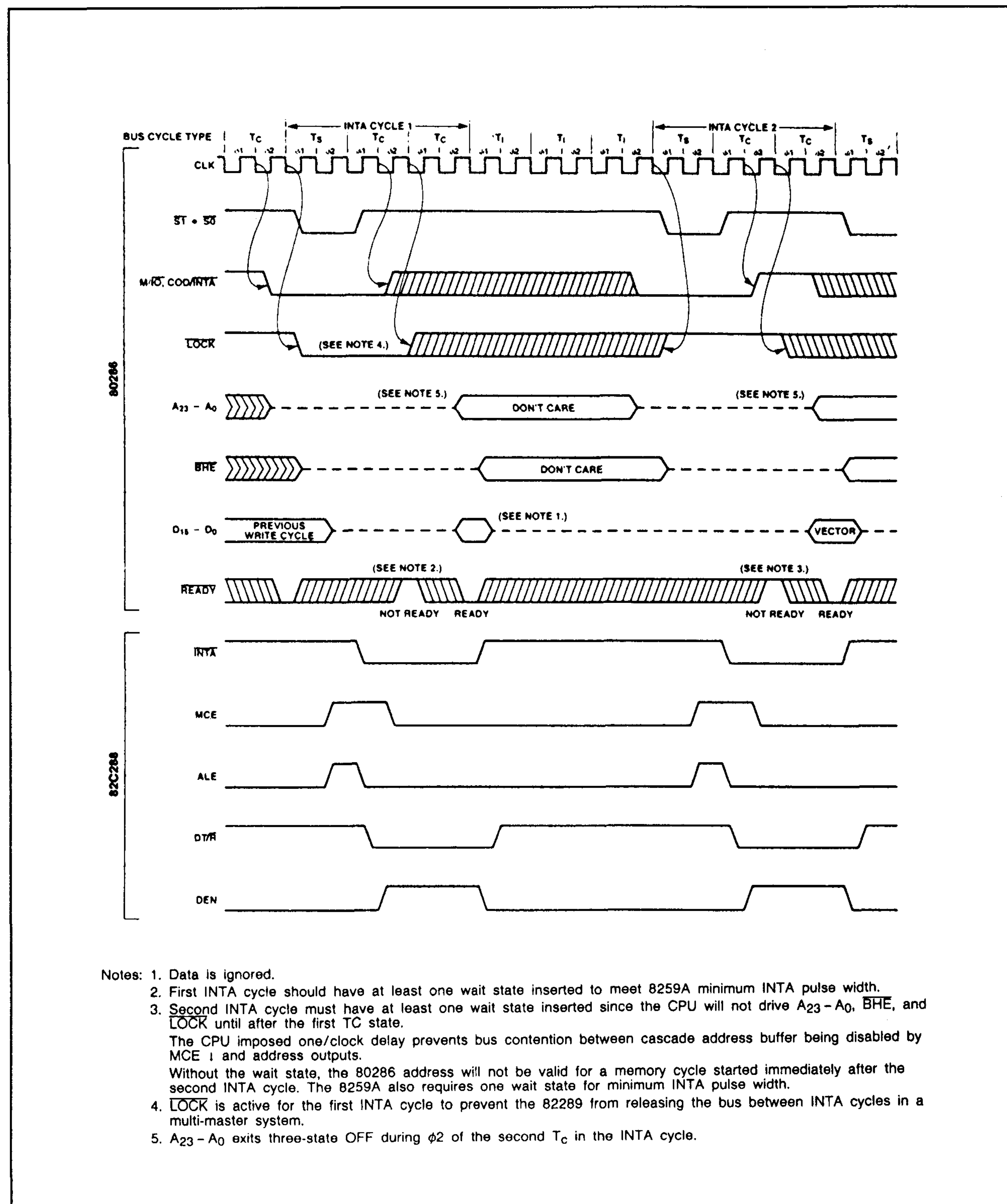
De 80286 Bus Unit (BU) zal reeds instructies ophalen voordat de lopende instructie wordt uitgevoerd. Deze aktiviteit wordt "prefetching" genoemd en treedt op als de lokale bus anders niet gebruikt zou zijn.
Er moet aan de volgende regels worden voldaan:

- Een prefetch busoperatie begint wanneer ten minste twee bytes in de 6-byte prefetch wachtrij (queue) leeg zijn.
- De prefetcher voert normaal woord-prefetches uit, onafhankelijk van de byte-uitrichting van de code-segment basis in het fysieke geheugen.
- De prefetcher voert alleen een byte code-fetch operatie uit voor besturingsoverdrachten naar een instructie die begint op een oneven fysiek adres.
- De prefetching eindigt wanneer een besturingsoverdracht of HLT-instructie door de IU wordt gedecodeerd en in de instructie wachtrij wordt geplaatst.
- In de reële adresseringsmode kan de prefetcher maximaal 6 bytes verder in een code-segment ophalen dan de laatste besturingsoverdracht of HLT instructie.
- In de beschermde mode zal de prefetcher nooit een segment-overrun uitzondering veroorzaken. De prefetcher stopt bij het laatste fysieke geheugenwoord van het code-segment. Als het programma probeert uit te voeren voorbij de laatste gehele instructie in het code-segment zal uitzondering 13 optreden. Als de laatste byte van een code-segment op een even geheugenadres verschijnt zal de prefetcher de volgende fysieke geheugenbyte uitlezen (een woord-code fetch uitvoeren). De waarde van deze byte wordt genegeerd en iedere poging hem uit te voeren veroorzaakt uitzondering 13.

Processor Extension Transfers

De processor uitbreidings-interface gebruikt de I/O-poortadressen 00F8(H), 00FA(H) en 00FC(H) die deel uitmaken van het voor I/O-poorten gereserveerde gebied.
Een ESC instructie met de MSW-bits EM = 0 en TS = 0 zal I/O-bus operaties uitvoeren op één of meer van deze I/O-poort adressen, onafhankelijk van de waarde van IOPL en CPL. ESC instructies met geheugenreferenties stellen de CPU in staat PEREQ-signalen voor processor-uitbreidings operandtransfers te accepteren. De CPU bepaalt het operand-startadres en de lees/schrijf-status van de instructie. Voor elke operand-transfer worden twee of drie busoperaties uitgevoerd: één woordoverdracht op I/O-poortadres 00FA(H) en één of twee busoperaties met geheugen. Drie busoperaties zijn nodig voor elke woordoperand die is uitgericht op een oneven byteadres.

3.3 80286



Figuur 7/3.3-31: Bevestiging van een verzoek tot onderbreking (Interrupt Acknowledge Sequentie).

3.3 80286

Let op!
Oneven uitgerichte numerieke operands dienen vermeden te worden wanneer een 80286 wordt gebruikt voor het runnen van zes of meer geheugen-wachttoestanden.
De 80286 kan een onjuist numeriek adres genereren als aan alle volgende voorwaarden wordt voldaan:
- De drijvende komma (floating point: FP) instructies zijn opgehaald en bevinden zich in de 80286 wachtrij.
- De eerste FP instructie is geen FSTSW AX.
- De tweede FP instructie gaat naar geheugen.
- De operand van de eerste instructie is uitgericht op een oneven geheugenadres.
- Er worden zes of meer wachttoestanden ingevoegd gedurende één van de laatste twee geheugen-schrijf transfers van de eerste instructie (oneven uitgerichte operands worden als twee bytes overgebracht).

Het adres van de tweede FP operand zal met één worden verhoogd als aan deze voorwaarden is voldaan. De kans dat deze voorwaarden optreden is het grootst in een multi-master systeem. Commando's naar de numerieke coprocessor moeten niet langer worden vertraagd dan met 9 T-toestanden. Bij grotere vertragingen (9 of meer) kan de synchronisatie tussen de 80286 en de 80287 worden verloren.

Interrupt Acknowledge Sequence
In figuur 7/3.3-31 wordt getoond hoe een onderbrekings-bevestiging (interrupt acknowledge sequence) door de 80286 wordt uitgevoerd als antwoord op een INTR-signaal. Deze volgorde bestaat uit twee INTA busoperaties. De eerste stelt een master 8259A Programmable Interrupt Controller (PIC) in staat te bepalen welke van zijn slaven (indien aanwezig) de interrupt-vector moet teruggeven. Tijdens de tweede INTA busoperatie wordt op D0 tot en met D7 van de 80286 een 8-bit vector gelezen om een interrupt handler-routine uit de interrupt-tabel te kiezen.

Het Master Cascade Enable (MCE) signaal van de 82288 wordt gebruikt om de cascade-adresdrivers gedurende de INTA busoperaties op de lokale adresbus aan te sluiten (figuur 7/3.3-31) om de slaaf-interrupt controllers via de systeem-adresbus te distribueren. De 80286 zendt tijdens T_S van de eerste INTA operatie het $\overline{LOCK}$-signaal uit (aktief-LAAG). Een lokaal bus-"hold" verzoek wordt pas aan het einde van de tweede INTA busoperatie gehonoreerd. De 80286 levert drie leegloop processorclocks tussen de INTA operaties om tegemoet te komen aan de minimum INTA-naar-INTA tijd en CAS (cascade adres) uitgangsvertraging van de 8259A.
De tweede INTA busoperatie moet altijd ten minste één extra T_C-toestand hebben die wordt toegevoegd door logische besturing van $\overline{READY}$. Dit is nodig om tegemoet te komen aan de minimale INTA pulsbreedte van de 8259A.

Prioriteiten bij lokaal busgebruik
Het gebruik van de lokale bus van de 80286 wordt gedeeld door verschillende interne units en externe HOLD requests.
Wanneer gelijktijdig om bezit van de bus wordt verzocht, zijn de relatieve prioriteiten:
- Hoogste
  - Transfers die $\overline{LOCK}$ expliciet (via de LOCK instructie-prefix) of impliciet aanspreken (bijvoorbeeld bij sommige segment-descriptor toegangen, interrupt acknowledge volgorde of een XCHG met geheugen).
  - De tweede van de twee-byte busoperaties die nodig zijn voor een oneven uitgerichte woord-operand.
  - De tweede of derde cyclus van een processor-uitbreidings data-transfer.
  - Lokale busrequest via de HOLD-ingang.
  - Processor-uitbreidings data-operand transfer via de PEREQ-ingang.

  - Data-transfer, uitgevoerd door EU, als deel van een instructie.
- Laagste
  - Een instructie-prefetch request door de BU. De EU zal prefetching twee processorclocks vóór een data-transfer verbieden om de EU zo kort mogelijk te laten wachten op het einde van een prefetch.

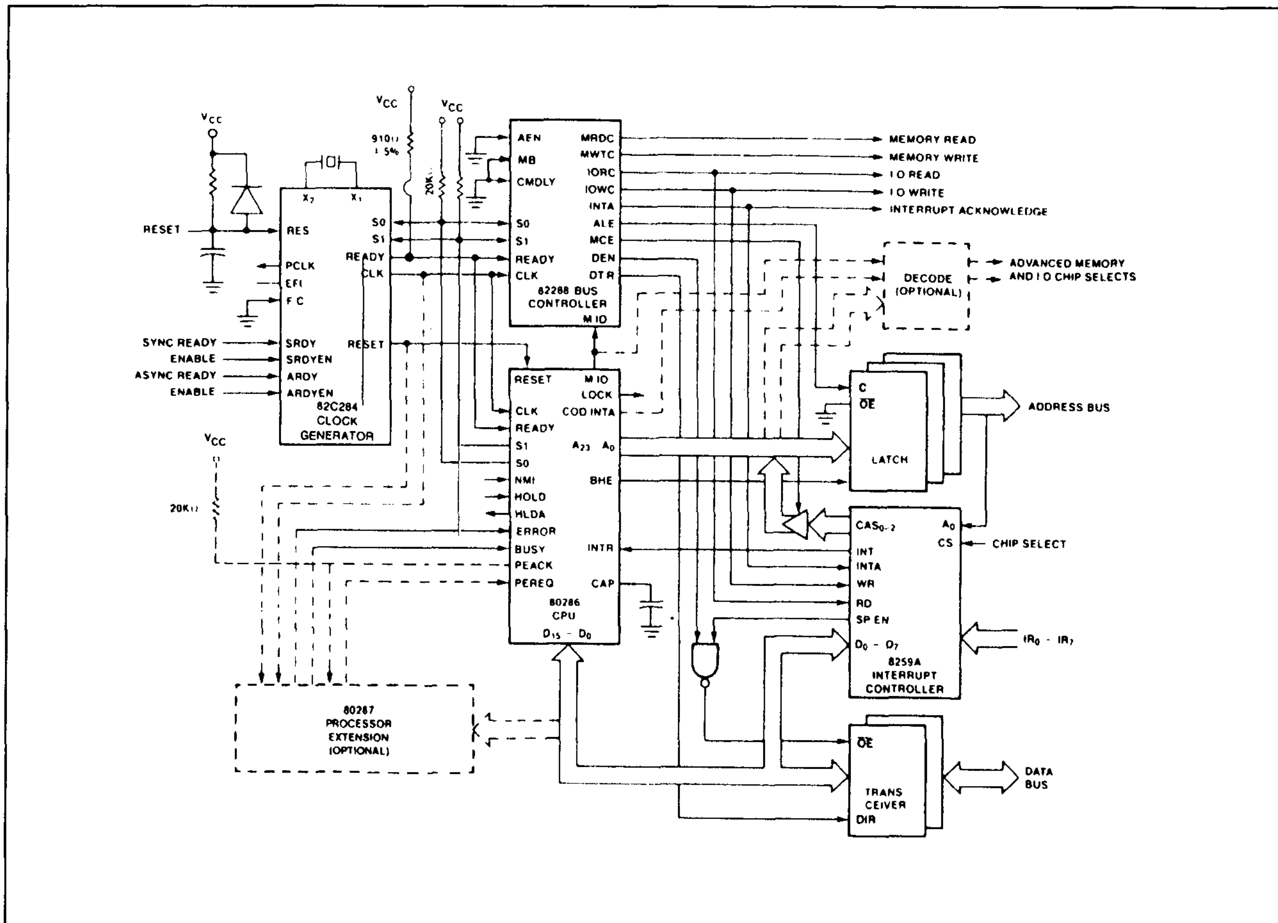
Halt of Shutdown Cycli
De 80286 geeft halt of shutdown condities extern aan als een busoperatie. Deze condities worden veroorzaakt door een HLT instructie of meerdere beveiligings-uitzonderingen terwijl wordt geprobeerd een instructie uit te voeren.
Een halt of shutdown busoperatie wordt gesignaleerd wanneer $\overline{S1}$, $\overline{S0}$ en COD/$\overline{INTA}$ LAAG zijn en M/$\overline{IO}$ HOOG is. Halt wordt aangegeven met A1 = HOOG en shutdown met A1 = LAAG. De 82288 buscontroller geeft geen ALE, terwijl ook geen $\overline{READY}$ nodig is om een einde te maken aan een halt of shutdown busoperatie.
Tijdens halt of shutdown kan de 80286 PEREQ of HOLD requests bedienen. Door een processor-uitbreidings segment-overrun uitzondering tijdens shutdown wordt verdere bediening van PEREQ gesperd. De 80286 wordt door NMI of RESET uit de halt of shutdown toestand gehaald. Een INTR (als de interrupts zijn vrijgegeven) of een processor-uitbreidings segment-overrun uitzondering haalt de 80286 ook uit halt.

Systeem configuraties

Inleiding
De veelzijdige busstructuur van het 80286 microsysteem maakt, in samenwerking met ondersteunende chips, flexibele configuratie van vele verschillende systemen mogelijk. De basis-configuratie die in figuur 7/3.3-32 te zien is, lijkt op een 8086 maximum modesysteem. Er wordt gebruik gemaakt van de CPU plus een 8259A interrupt-controller, een 82(C)284 clock-generator en een 82288 bus-controller.

Met de stippellijnen wordt in figuur 7/3.3-32 aangegeven dat gemakkelijk uitbreidingen kunnen worden aangebracht. De processor uitbreidings-interface stelt externe hardware in staat speciale functies te verrichten en data gelijktijdig over te brengen met CPU-executies of andere instructies. Volledige systeem-integriteit blijft gehandhaafd omdat de 80286 toezicht houdt op alle data-overdrachten en uitvoeringen van instructies voor de processor-extensie. De 80287 heeft alle instructies en data-typen van een 8087. De 80287 NPX kan numerieke berekeningen en data-transfers samenvallend met CPU programma-executies uitvoeren. Numerieke code en data hebben dezelfde integriteit als alle andere informatie die beveiligd wordt door het 80286 protectie-mechanisme.
De 80286 kan chip-select decodering en adres-verwerking overlappend laten plaatsvinden tijdens data-overdracht voor de vorige busoperatie. Deze informatie wordt door ALE middenin een T_S-cyclus gelatcht. De gelatchte chip-select en adres-informatie blijven gedurende de busoperatie stabiel, terwijl het adres van de volgende cyclus wordt gedecodeerd en in het systeem gebracht. Voor de decodeerlogika kan een snelle bipolaire PROM worden toegepast. De optionele decodeerlogika in figuur 7/3.3-32 maakt gebruik van het overlappen van adres en data bij de 80286 buscyclus voor het genereren van geavanceerde geheugen- en IO-select signalen. Hierdoor wordt degradatie van het systeem door vertragingen tot een minimum beperkt. Behalve voor het selecteren van geheugen en I/O mogen de geavanceerde select-signalen ook worden gebruikt voor configuraties die lokale en systeem-bussen ondersteunen om deze in staat te stellen bij elke buscyclus de juiste bus-interface vrij te geven. De COD/$\overline{INTA}$ en M/$\overline{IO}$ signalen worden aan de decodeerlogika toegevoerd om onderscheid te kunnen maken tussen interruptie-, I/O-, code- en databus cycli.

3.3 80286



Figuur 7/3.3-32: Basis 80286 systeemconfiguratie.

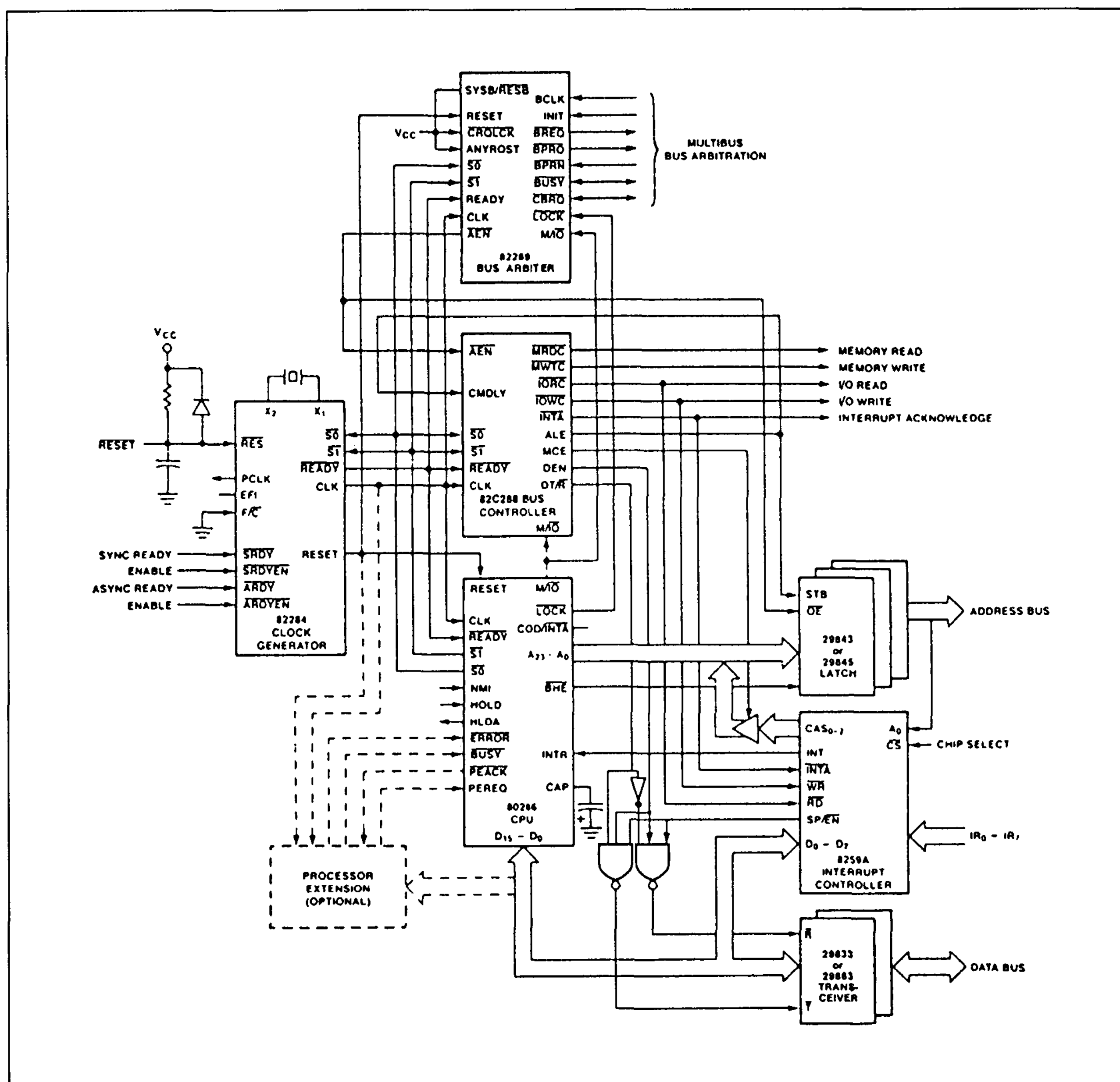
Door toevoegen van de 82289 bus-arbiter chip levert de 80286 een Multibus systeembus-interface op, zoals in figuur 7/3.3-33 wordt getoond. De ALE-uitgang van de 82288 voor de Multibus wordt verbonden met zijn CMDLY-ingang om de start van commando's één systeem-CLK te vertragen, hetgeen nodig is om tegemoet te komen aan de setup tijden voor adres en schrijf-data van de Multibus. Deze opstelling zal ten minste één extra T_C-toestand toevoegen aan elke busoperatie die gebruik maakt van de Multibus.

Aan de lokale bus in figuur 7/3.3-33 zouden nog een tweede 82288 buscontroller en extra latches en transceivers kunnen worden toegevoegd.

Hierdoor kan de 80286 dan een on-board bus voor lokaal geheugen en periferieschakelingen ondersteunen plus de Multibus voor systeembus-interfaces.

In figuur 7/3.3-34 is dual-port dynamisch geheugen (RAM met twee data-poorten) tussen de Multibus systeembus en de 80286 lokale bus geplaatst. De dual-port interface wordt geleverd door de 8207 Dual Port DRAM Controller. De 8207 werkt synchroon met de CPU om de verwerkingstijd bij lokale geheugen-toegangen zo kort mogelijk te houden.
Hij verzorgt ook de arbitrage bij verzoeken van de lokale- en systeembussen en voert functies uit zoals refresh en initialisatie van RAM en read/modify/write cycli.

3.3 80286



Figuur 7/3.3-33: Een Multibus systeembus-interface.

De 8207 kan in combinatie met de 8206 Error Checking and Correction geheugencontroller enkele bits corrigeren.
Het dual-port geheugen kan worden gecombineerd met een standaard Multibus systeembus-interface voor maximale prestaties en beveiligingen in multi-processor systeemconfiguraties.

Elektrische en timing kenmerken

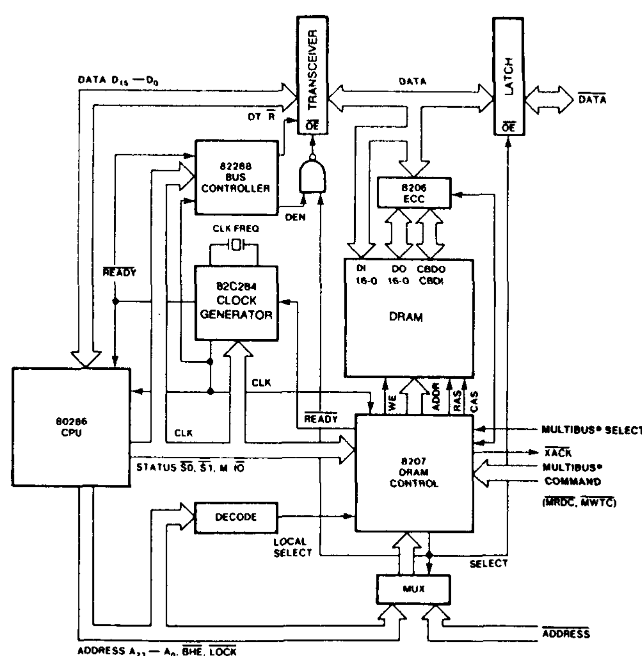
In de tabellen 7/3.3-24 tot en met 7/3.3-29 en de figuren 7/3.3-35 tot en met 7/3.3-40 zijn de elektrische en timing karakteristieken van de 80286 samengevat.

3.3 80286

80286 Pin and Name	Pullup Value	Purpose
4—$\overline{S1}$	20 KΩ ±10%	Pull $\overline{S0}$, $\overline{S1}$, and $\overline{PEACK}$ inactive during 80286 hold periods(1)
5—$\overline{S0}$		
6—$\overline{PEACK}$		
63—$\overline{READY}$	910Ω ±5%	Pull $\overline{READY}$ inactive within required minimum time (C_L = 150 pF, $I_R \leq$ 7 mA)

NOTE:
1. Pull-up resistors are not required on $\overline{S0}$ and $\overline{S1}$ when the corresponding pins of the 82C284 are connected to $\overline{S0}$ and $\overline{S1}$.

Tabel 7/3.3-23: Aanbevolen waarden voor de in een 80286 systeem benodigde optrekweerstanden.

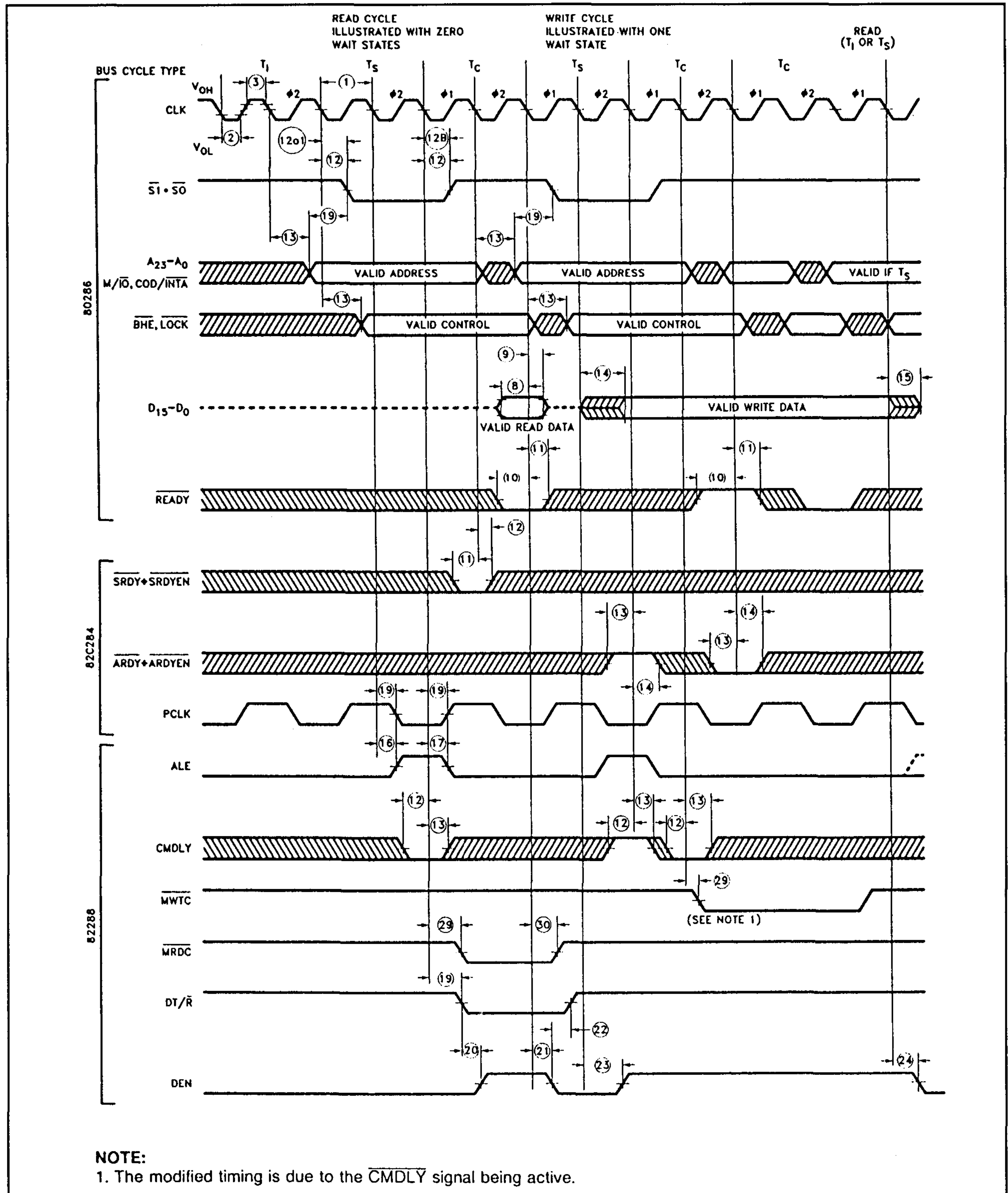


Figuur 7/3.3-34: Uitbreiding van een 80286 systeem met een dual-port geheugen.

Ambient Temperature Under Bias .... 0°C to + 70°C
Storage Temperature .... −65°C to + 150°C
Voltage on Any Pin with
Respect to Ground .... − 1.0V to + 7V
Power Dissipation 3.3W

Tabel 7/3.3-24: Maximaal toegelaten waarden.

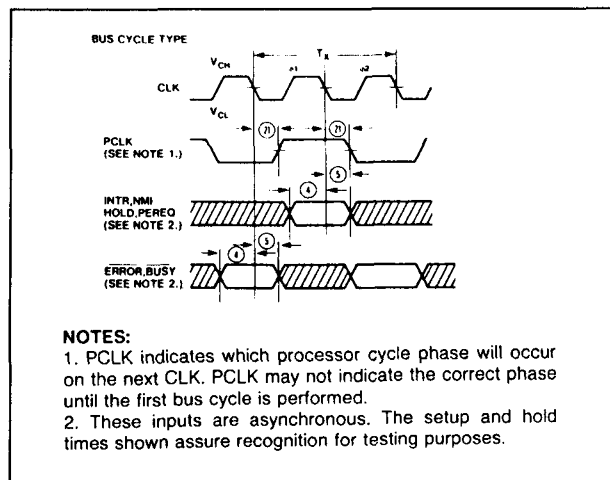
3.3 80286



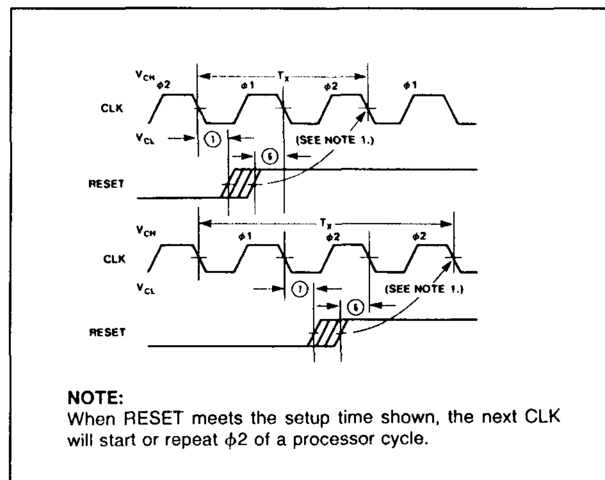
NOTE:
1. The modified timing is due to the CMDLY signal being active.

Figuur 7/3.3-35: Golfvormen en samenhang van de schakeltijden voor de 80286 (en 82284 en 82288).

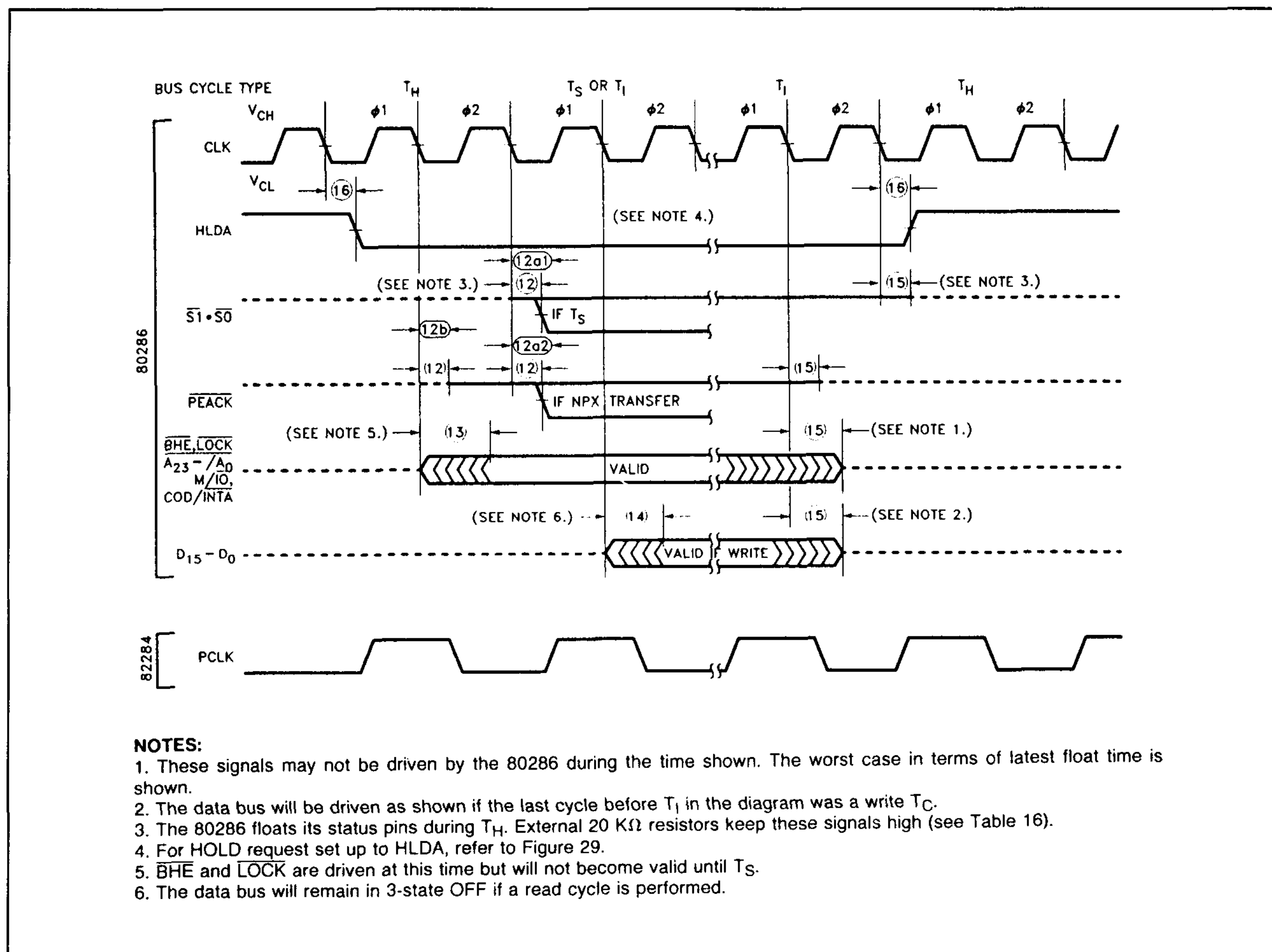
3.3 80286



Figuur 7/3.3-36: Timing van asynchrone ingangssignalen voor de 80286.



Figuur 7/3.3-37: Timing van het RESET-signaal en daaropvolgende fase van de processorcyclus.



Figuur 7/3.3-38: Opwekken van en binnentreden in HOLD.

3.3 80286

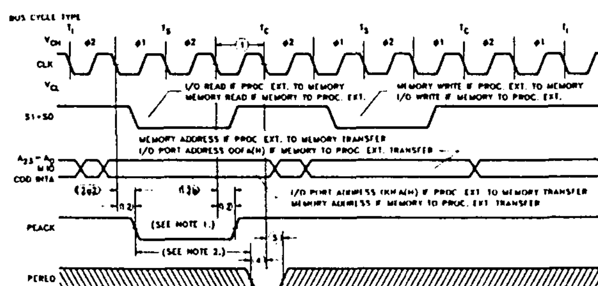
Symbol	Parameter	Min	Max	Unit
I_{CC}	Supply Current (0°C Turn On)		600	mA
C_{CLK}	CLK Input Capacitance		20	pF
C_{IN}	Other Input Capacitance		10	pF
C_O	Input/Output Capacitance		20	pF

Tabel 7/3.3-25: Gelijkspannings-karakteristieken (voedingsstroom en capaciteiten).

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input LOW Voltage	–0.5	0.8	V	
V_{IH}	Input HIGH Voltage	2.0	V_{CC} + 0.5	V	
V_{ILC}	CLK Input LOW Voltage	–0.5	0.6	V	
V_{IHC}	CLK Input HIGH Voltage	3.8	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltge		0.45	V	I_{OL} = 2.0 mA
V_{OH}	Output HIGH Voltage	2.4		V	I_{OH} = –400.0 μA
I_{LI}	Input Leakage Current		± 10	μA	$0V \leq V_{IN} \leq V_{CC}$
I_{LCR}	Input CLK, RESET Leakage Current		± 10	μA	$0.45 \leq V_{IN} \leq V_{CC}$
I_{LCR}	Input CLK, RESET Leakage Current		± 1	mA	$0 \leq V_{IN} < 0.45$
I_{IL}	Input Sustaining Current on BUSY and ERROR Pins	30	500	μA	V_{IN} = 0V
I_{LO}	Output Leakage Current		± 10	μA	$0.45 \leq V_{OUT} \leq V_{CC}$
I_{LO}	Output Leakage Current		± 1	mA	$0 \leq V_{OUT} < 0.45$

*T_A is guaranteed from 0°C to + 55°C as long as T_{CASE} is not exceeded.

Tabel 7/3.3-26: Overige gelijkspannings-karakteristieken.



NOTES:

1. PEACK always goes active during the first bus operation of a processor extension data operand transfer sequence. The first bus operation will be either a memory read at operand address or I/O read at port address OOFA(H).
2. To prevent a second processor extension data operand transfer, the worst case maximum time (Shown above) is: 3× ① – 12a$_{2max.}$ – ④$_{min.}$. The actual, configuration dependent, maximum time is: 3× ① – 12a$_{2max.}$ – ④$_{min.}$ + A×2× ①.
A is the number of extra T_C states added to either the first or second bus operation of the processor extension data operand transfer sequence.

Figuur 7/3.3-39: Timing van PEREQ/PEACK voor slechts één transfer.

3.3 80286

AC timings are referenced to 0.8V and 2.0V points of signals as illustrated in datasheet waveforms, unless otherwise noted.

Symbol	Parameter	6 MHz		8 MHz		10 MHz		12.5 MHz (Preliminary)		Unit	Test Condition
		-6 Min	-6 Max	-8 Min	-8 Max	-10 Min	-10 Max	-12 Min	-12 Max		
1	System Clock (CLK) Period	83	250	62	250	50	250	40	250	ns	
2	System Clock (CLK) LOW Time	20	225	15	225	12	232	11	237	ns	at 1.0V
3	System Clock (CLK) HIGH Time	25	230	25	235	16	239	13	239	ns	at 3.6V
17	System Clock (CLK) Rise Time		10		10		8	—	8	ns	1.0V to 3.6V, (Note 7)
18	System Clock (CLK) Fall Time		10		10		8	—	8	ns	3.6V to 1.0V, (Note 7)
4	Asynch. Inputs Setup Time	30		20		20		15		ns	(Note 1)
5	Asynch. Inputs Hold Time	30		20		20		15		ns	(Note 1)
6	$\overline{\text{RESET}}$ Setup Time	33		28		23		18		ns	
7	$\overline{\text{RESET}}$ Hold Time	5		5		5		5		ns	
8	Read Data Setup Time	20		10		8		5		ns	
9	Read Data Hold Time	8		8		8		6		ns	
10	$\overline{\text{READY}}$ Setup Time	50		38		26		22		ns	
11	$\overline{\text{READY}}$ Hold Time	35		25		25		20		ns	
12	Status/$\overline{\text{PEACK}}$ Valid Delay	1	55	1	40	—	—	—	—	ns	(Notes 2, 3)
12a1	Status Active Delay	—	—	—	—	1	22	3	18	ns	(Notes 2, 3)
12a2	$\overline{\text{PEACK}}$ Active Delay	—	—	—	—	1	22	3	20	ns	(Notes 2, 3)
12b	Status/$\overline{\text{PEACK}}$ Inactive Delay	—	—	—	—	1	30	3	22	ns	(Notes 2, 3)
13	Address Valid Delay	1	80	1	60	1	35	1	32	ns	(Notes 2, 3)
14	Write Data Valid Delay	0	65	0	50	0	30	0	30	ns	(Notes 2, 3)
15	Address/Status/Data Float Delay	0	80	0	50	0	47	0	32	ns	(Notes 2, 4, 7)
16	HLDA Valid Delay	0	80	0	50	0	47	0	27	ns	(Notes 2, 3)
19	Address Valid To Status Valid Setup Time	—		38		27		22		ns	(Notes 3, 5, 6)

*T_A is guaranteed from 0°C to +55°C as long as T_{CASE} is not exceeded.

NOTES:
1. Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.
2. Delay from 1.0V on the CLK, to 0.8V or 2.0V or float on the output as appropriate for valid or floating condition.
3. Output load: C_L = 100 pF.
4. Float condition occurs when output current is less than I_{LO} in magnitude.
5. Delay measured from address either reaching 0.8V or 2.0V (valid) to status going active reaching 2.0V or status going inactive reaching 0.8V.
6. For load capacitance of 10 pF or more on STATUS/PEACK lines, subtract typically 7 ns for 8 MHz, 10 MHz and 12.5 MHz spec.
7. These are not tested. They are guaranteed by design characterization.

Tabel 7/3.3-27: AC-karakteristieken (schakeltijden van de 80286).

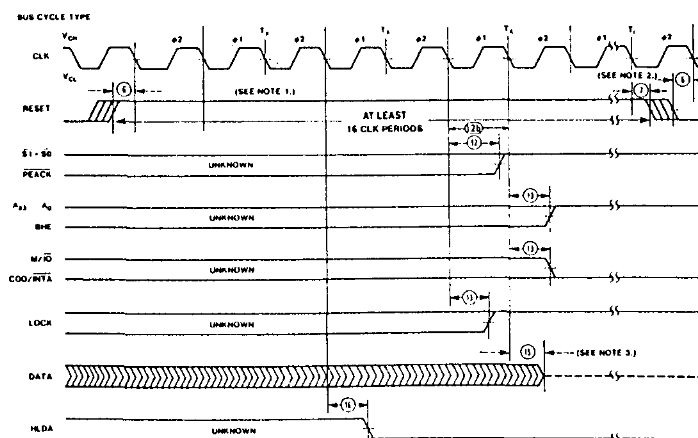
Symbol	Parameter	82C284-6 (Advance)		82C284-8 (Advance)		82C284-10 (Advance)		82C284-12 (Advance)		Units	Test Conditions
		Min	Max	Min	Max	Min	Max	Min	Max		
11	$\overline{\text{SRDY}}$/$\overline{\text{SRDYEN}}$ Setup Time	25		17		15		15		ns	
12	$\overline{\text{SRDY}}$/$\overline{\text{SRDYEN}}$ Hold Time	0		0		2		2		ns	
13	$\overline{\text{ARDY}}$/$\overline{\text{ARDYEN}}$ Setup Time	5		0		0		0		ns	(Note 1)
14	$\overline{\text{ARDY}}$/$\overline{\text{ARDYEN}}$ Hold Time	30		30		30		25		ns	(Note 1)
19	PCLK Delay	0	45	0	45	0	35	0	23	ns	C_L = 75 pF I_{OL} = 5 mA I_{OH} = −1 mA

Tabel 7/3.3-28: Eisen die aan de timing van de 82(C)284 clock-generator worden gesteld (zie ook figuur 7/3.3-35).

3.3 80286

Symbol	Parameter		82288-6 Min	82288-6 Max	82288-8 Min	82288-8 Max	82288-10 Min	82288-10 Max	82288-12 (Preliminary) Min	82288-12 (Preliminary) Max	Units	Test Conditions
12	CMDLY Setup Time		25		20		15		15		ns	
13	CMDLY Hold Time		1		1		1		1		ns	
30	Command Delay from CLK	Command Inactive	5	30	5	20	5	20	5	20	ns	C_L = 300 pF max, I_{OL} = 32 mA max, I_{OH} = −5 mA max
29		Command Active	3	40	3	25	3	21	3	21		
16	ALE Active Delay		3	25	3	20	3	16	3	16	ns	C_L = 150 pF, I_{OL} = 16 mA max, I_{OH} = −1 mA max
17	ALE Inactive Delay			35		25		19		19	ns	
19	DT/$\overline{R}$ Read Active Delay			40		25		23		23	ns	
22	DT/$\overline{R}$ Read Inactive Delay		5	45	5	35	5	20	5	18	ns	
20	DEN Read Active Delay		5	50	5	35	5	21	5	21	ns	
21	DEN Read Inactive Delay		3	40	3	35	3	21	3	19	ns	
23	DEN Write Active Delay			35		30		23		23	ns	
24	DEN Write Inactive Delay		3	35	3	30	3	19	3	19	ns	

Tabel 7/3.3-29: Timingvereisten voor de 82288 buscontroller (zie ook figuur 7/3.3-35).



NOTES:
1. Setup time for RESET ↑ may be violated with the consideration that ϕ1 of the processor clock may begin one system CLK period later.
2. Setup and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during ϕ1 or ϕ2.
3. The data bus is only guaranteed to be in 3-state OFF at the time shown.

Figuur 7/3.3-40: Initiële pen-toestanden van de 80286 tijdens RESET.

Samenvatting van de 80286 instructieset

Timing van de instructies

De maximale uitvoeringssnelheid van de 80286 wordt bepaald door de aantallen benodigde clock-pulsen (clock count). Als er geen vertragingen in de buscycli optreden, zal het werkelijke aantal clockpulsen voor een 80286 programma ongeveer 5 % hoger zijn dan het uitgerekende aantal vanwege instructie-volgorden die sneller worden uitgevoerd dan zij uit geheugen kunnen worden opgehaald.

Om de voor instructie-volgorden benodigde tijden uit te rekenen moet de som van alle instructie clock-counts worden vermenigvuldigd met de clock-periode van de processor. Een 8 MHz processorclock heeft een clockperiode van 125 ns, waarvoor een systeemclock (CLK-ingang) nodig is van 16 MHz.

Aannamen voor Instructie-clock counts

- De instructie is van te voren opgehaald (prefetched), gedecodeerd en klaar voor uitvoering. Control transfer-instructie clock counts zijn inclusief alle tijd die nodig is voor ophalen, decoderen en prepareren voor uitvoering van de volgende instructie.
- Buscycli hebben geen wait-states nodig.
- Er zijn geen processor-uitbreidings data-transfer of lokale bus HOLD verzoeken.
- Tijdens uitvoering van de instructie treden geen uitzonderingen op.

Notities bij de samenvatting van de Instructieset

- Door het MOD-veld geselecteerde relatieve adressen worden niet getoond. Indien noodzakelijk verschijnen zij na de getoonde instructievelden.
- Above/below heeft betrekking op waarden zonder teken.
- Greater heeft betrekking op waarden met positief teken.
- Less heeft betrekking op minder positieve (meer negatieve) waarden.
- Als d = 1: naar register; als d = 0: van register.
- als w = 1: woord-instructie; als w = 0: byte-instructie.
- als s = 0: 16-bit immediate data uit de operand; als s = 1: wordt een immediate databyte met teken uitgebreid om een 16-bit operand te vormen.
- x: maakt niet uit (don't care).
- z: wordt gebruikt voor string-primitieven voor vergelijking met de ZF flag.

Als er twee clock-counts worden opgegeven, heeft de kleinste betrekking op een register-operand en de grootste op een geheugen-operand.

- *: voeg één clock toe als voor de offset berekening 3 elementen nodig zijn.
- n: aantal herhaalde tijden.
- m: aantal code-bytes in volgende instructie.
- Level (L): lexicografisch nesting-niveau van de procedure.

Toelichtingen

De volgende commentaren beschrijven mogelijke uitzonderingen, zijeffecten en toegestaan gebruik voor instructies in beide bedrijfsmodes van de 80286 (zie ook de instructieset, tabel 7/3.3-30).

Alleen reële adresmode

- 1
  Dit is een protected mode instructie. Pogingen deze uit te voeren in de reële adresseringsmode zal een ongedefinieerde opcode-uitzondering (6) tot gevolg hebben.
- 2
  Een segment overrun-uitzondering (13) treedt op als een woord operand-referentie bij offset FFFF(H) wordt geprobeerd.
- 3
  Deze instructie mag in de reële adresseringsmode worden uitgevoerd om de CPU op de beschermde mode voor te bereiden.
- 4
  De IOPL en NT velden zullen 0 blijven.
- 5
  Als de operand de segment-limiet overschreidt zal een processor-uitbreidings segment-overrun-interruptie (9) optreden.

Beide modes

- 6
  Een uitzondering kan optreden, afhankelijk van de waarde van de operand.
- 7
  $\overline{\text{LOCK}}$ wordt automatisch opgelegd, onafhankelijk van de aan- of afwezigheid van de LOCK instructie-prefix.
- 8
  $\overline{\text{LOCK}}$ blijft niet aktief tussen alle operand-transfers.

Alleen beschermde virtuele adresmode

- 9
  Een algemene beveiligings-uitzondering (13) zal optreden als de geheugen-operand niet gebruikt kan worden door een segment-limiet of overtreding van toe-

3.3 80286

gangsrechten. Als een stack segmentlimiet wordt geschonden, treedt een stack-segment overrun-uitzondering (12) op.

- 10
  Voor segment load-operaties moeten de CPL, RPL en DPL het eens zijn over de privilege-regels om een uitzondering te vermijden.
  Als het segment niet aanwezig is treedt een niet-aanwezig uitzondering (11) op. Als het SS register de bestemming is en een segment niet-aanwezig overtreding optreedt, verschijnt een stack uitzondering (12).
- 11
  Alle segment-descriptor toegangen in de GDT of LDT die door deze instructie worden gedaan, aktiveren $\overline{LOCK}$ automatisch om descriptor-integriteit in multi-processor systemen te handhaven.
- 12
  JMP, CALL, INT, RET en IRET instructies die op een ander codesegment betrekking hebben zullen een algemene beveiligings-uitzondering (13) veroorzaken als een privilege-regel wordt geschonden.
- 13
  Als CPL = geen 0 treedt een algemene beveiligings-uitzondering (13) op.
- 14
  Als CPL > IOPL treedt een algemene beveiligings-uitzondering (13) op.
- 15
  Het IF-veld van het flag-woord is niet op orde gebracht als CPL > IOPL.
  Het IOPL-veld wordt alleen geaktualiseerd als CPL = 0.
- 16
  Overtredingen van privilege-regels die van toepassing zijn op de selector-operand veroorzaken geen beveiligings-uitzondering; de instructie geeft echter geen resultaat terug en de zero-flag wordt gecleared.
- 17
  Als het startadres van de geheugen-operand een segment-limiet overtreedt of als een ongeldige toegang wordt geprobeerd, treedt een algemene beveiligings-uitzondering (13) op voordat de ESC instructie is uitgevoerd.
  Een stack-segment overrun-uitzondering (12) treedt op als de stack-limiet door het startadres van de operand wordt geschonden.
  Als tijdens een poging tot data-transfer een segment-limiet wordt geschonden treedt een processor-uitbreidings segment-overrun-uitzondering (9) op.
- 18
  De bestemming van een INT, JMP, CALL, RET of IRET instructie moet binnen de gedefinieerde limiet van een codesegment blijven of er zal een algemene beveiligings-uitzondering (13) optreden.

Overzicht

In de tabellen 7/3.3-30a tot en met 7/3.3-30h wordt een compleet overzicht gegeven van de 80286 instructieset, terwijl tabel in 7/3.3-31 nog enkele aanwijzingen voor gebruik staan.
De instructies die op een donkere achtergrond zijn afgedrukt bestaan niet voor 8086/8088 microsystemen.

3.3 80286

FUNCTION	FORMAT				CLOCK COUNT		COMMENTS	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
DATA TRANSFER								
MOV = Move:								
Register to Register/Memory	1000100w	mod reg r/m			2,3*	2,3*	2	9
Register/memory to register	1000101w	mod reg r/m			2,5*	2,5*	2	9
Immediate to register/memory	1100011w	mod 000 r/m	data	data if w = 1	2,3*	2,3*	2	9
Immediate to register	1011w reg	data	data if w = 1		2	2		
Memory to accumulator	1010000w	addr-low	addr-high		5	5	2	9
Accumulator to memory	1010001w	addr-low	addr-high		3	3	2	9
Register/memory to segment register	10001110	mod 0 reg r/m			2,5*	17,19*	2	9,10,11
Segment register to register/memory	10001100	mod 0 reg r/m			2,3*	2,3*	2	9
PUSH = Push:								
Memory	11111111	mod 110 r/m			5*	5*	2	9
Register	01010 reg				3	3	2	9
Segment register	000reg110				3	3	2	9
Immediate	011010s0	data	data if s = 0		3	3	2	9
PUSHA = Push All	01100000				17	17	2	9
POP = Pop:								
Memory	10001111	mod 000 r/m			5*	5*	2	9
Register	01011 reg				5	5	2	9
Segment register	000reg111	(reg≠01)			5	20	2	9,10,11
POPA = Pop All	01100001				19	19	2	9
XCHG = Exhcange:								
Register/memory with register	1000011w	mod reg r/m			3,5*	3,5*	2,7	7,9
Register with accumulator	10010 reg				3	3		
IN = Input from:								
Fixed port	1110010w	port			5	5		14
Variable port	1110110w				5	5		14
OUT = Output to:								
Fixed port	1110011w	port			3	3		14
Variable port	1110111w				3	3		14
XLAT = Translate byte to AL	11010111				5	5		9
LEA = Load EA to register	10001101	mod reg r/m			3*	3*		
LDS = Load pointer to DS	11000101	mod reg r/m	(mod≠11)		7*	21*	2	9,10,11
LES = Load pointer to ES	11000100	mod reg r/m	(mod≠1)		7*	21*	2	9,10,11

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Tabel 7/3.3-30a: Samenvatting van de 80286 instructieset, deel 1.

3.3 80286

FUNCTION	FORMAT				CLOCK COUNT		COMMENTS	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
DATA TRANSFER (Continued)								
LAHF Load AH with flags	10011111				2	2		
SAHF = Store AH into flags	10011110				2	2		
PUSHF = Push flags	10011100				3	3	2	9
POPF = Pop flags	10011101				5	5	2,4	9,15
ARITHMETIC								
ADD = Add:								
Reg/memory with register to either	000000dw	mod reg r/m			2,7*	2,7*	2	9
Immediate to register/memory	100000sw	mod 0 0 0 r/m	data	data if s w = 01	3,7*	3,7*	2	9
Immediate to accumulator	0000010w	data	data if w = 1		3	3		
ADC = Add with carry:								
Reg/memory with register to either	000100dw	mod reg r/m			2,7*	2,7*	2	9
Immediate to register/memory	100000sw	mod 0 1 0 r/m	data	data if s w = 01	3,7*	3,7*	2	9
Immediate to accumulator	0001010w	data	data if w = 1		3	3		
INC = Increment:								
Register/memory	1111111w	mod 0 0 0 r/m			2,7*	2,7*	2	9
Register	01000reg				2	2		
SUB = Subtract:								
Reg/memory and register to either	001010dw	mod reg r/m			2,7*	2,7*	2	9
Immediate from register/memory	100000sw	mod 1 0 1 r/m	data	data if s w = 01	3,7*	3,7*	2	9
Immediate from accumulator	0010110w	data	data if w = 1		3	3		
SBB = Subtract with borrow:								
Reg/memory and register to either	000110dw	mod reg r/m			2,7*	2,7*	2	9
Immediate from register/memory	100000sw	mod 0 1 1 r/m	data	data if s w = 01	3,7*	3,7*	2	9
Immediate from accumulator	0001110w	data	data if w = 1		3	3		
DEC = Decrement								
Register/memory	1111111w	mod 0 0 1 r/m			2,7*	2,7*	2	9
Register	01001 reg				2	2		
CMP = Compare								
Register/memory with register	0011101w	mod reg r/m			2,6*	2,6*	2	9
Register with register/memory	0011100w	mod reg r/m			2,7*	2,7*	2	9
Immediate with register/memory	100000sw	mod 1 1 1 r/m	data	data if s w = 01	3,6*	3,6*	2	9
Immediate with accumulator	0011110w	data	data if w = 1		3	3		
NEG = Change sign	1111011w	mod 0 1 1 r/m			2	7*	2	9
AAA = ASCII adjust for add	00110111				3	3		
DAA = Decimal adjust for add	00100111				3	3		

Tabel 7/3.3-30b: Samenvatting van de 80286 instructieset, deel 2.

3.3 80286

FUNCTION	FORMAT	CLOCK COUNT Real Address Mode	CLOCK COUNT Protected Virtual Address Mode	COMMENTS Real Address Mode	COMMENTS Protected Virtual Address Mode
ARITHMETIC (Continued)					
AAS = ASCII adjust for subtract	00111111	3	3		
DAS = Decimal adjust for subtract	00101111	3	3		
MUL = Multiply (unsigned):	1111011w | mod 100 r/m				
Register-Byte		13	13		
Register-Word		21	21		
Memory-Byte		16*	16*	2	9
Memory-Word		24*	24*	2	9
IMUL = Integer multiply (signed):	1111011w | mod 101 r/m				
Register-Byte		13	13		
Register-Word		21	21		
Memory-Byte		16*	16*	2	9
Memory-Word		24*	24*	2	9
IMUL = Integer immediate multiply (signed)	011010s1 | mod reg r/m | data | data if s = 0	21,24*	21,24*	2	9
DIV = Divide (unsigned)	1111011w | mod 110 r/m				
Register-Byte		14	14	6	6
Register-Word		22	22	6	6
Memory-Byte		17*	17*	2,6	6,9
Memory-Word		25*	25*	2,6	6,9
IDIV = Integer divide (signed)	1111011w | mod 111 r/m				
Register-Byte		17	17	6	6
Register-Word		25	25	6	6
Memory-Byte		20*	20*	2,6	6,9
Memory-Word		28*	28*	2,6	6,9
AAM = ASCII adjust for multiply	11010100 | 00001010	16	16		
AAD = ASCII adjust for divide	11010101 | 00001010	14	14		
CBW = Convert byte to word	10011000	2	2		
CWD = Convert word to double word	10011001	2	2		
LOGIC					
Shift/Rotate Instructions:					
Register/Memory by 1	1101000w | mod TTT r/m	2,7*	2,7*	2	9
Register/Memory by CL	1101001w | mod TTT r/m	5+n,8+n*	5+n,8+n*	2	9
Register/Memory by Count	**1100000w | mod TTT r/m | count**	**5+n,8+n***	**5+n,8+n***	**2**	**9**

TTT	Instruction
000	ROL
001	ROR
010	RCL
011	RCR
100	SHL/SAL
101	SHR
111	SAR

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Tabel 7/3.3-30c: Samenvatting van de 80286 instructieset, deel 3.

3.3 80286

FUNCTION	FORMAT				CLOCK COUNT		COMMENTS	
					Real Address Mode	Protected Virtual Address Mode	Real Address Mode	Protected Virtual Address Mode
ARITHMETIC (Continued)								
AND = **And:**								
Reg/memory and register to either	001000dw	mod reg r/m			2,7*	2,7*	2	9
Immediate to register/memory	1000000w	mod 1 0 0 r/m	data	data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0010010w	data	data if w = 1		3	3		
TEST = **And function to flags, no result:**								
Register/memory and register	1000010w	mod reg r/m			2,6*	2,6*	2	9
Immediate data and register/memory	1111011w	mod 0 0 0 r/m	data	data if w = 1	3,6*	3,6*	2	9
Immediate data and accumulator	1010100w	data	data if w = 1		3	3		
OR = **Or:**								
Reg/memory and register to either	000010dw	mod reg r/m			2,7*	2,7*	2	9
Immediate to register/memory	1000000w	mod 0 0 1 r/m	data	data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0000110w	data	data if w = 1		3	3		
XOR = **Exclusive or:**								
Reg/memory and register to either	001100dw	mod reg r/m			2,7*	2,7*	2	9
Immediate to register/memory	1000000w	mod 1 1 0 r/m	data	data if w = 1	3,7*	3,7*	2	9
Immediate to accumulator	0011010w	data	data if w = 1		3	3		
NOT = Invert register/memory	1111011w	mod 0 1 0 r/m			2,7*	2,7*	2	9
STRING MANIPULATION:								
MOVS = Move byte/word	1010010w				5	5	2	9
CMPS = Compare byte/word	1010011w				8	8	2	9
SCAS = Scan byte/word	1010111w				7	7	2	9
LODS = Load byte/wd to AL/AX	1010110w				5	5	2	9
STOS = Stor byte/wd from AL/A	1010101w				3	3	2	9
INS = **Input byte/wd from DX port**	0110110w				5	5	2	9,14
OUTS = **Output byte/wd to DX port**	0110111w				5	5	2	9,14
Repeated by count in CX								
MOVS = Move string	11110011	1010010w			5 + 4n	5 + 4n	2	9
CMPS = Compare string	1111001z	1010011w			5 + 9n	5 + 9n	2,8	8,9
SCAS = Scan string	1111001z	1010111w			5 + 8n	5 + 8n	2,8	8,9
LODS = Load string	11110011	1010110w			5 + 4n	5 + 4n	2,8	8,9
STOS = Store string	11110011	1010101w			4 + 3n	4 + 3n	2,8	8,9
INS = **Input string**	11110011	0110110w			5+4n	5+4n	2	9,14
OUTS = **Output string**	11110011	0110111w			5+4n	5+4n	2	9,14

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Tabel 7/3.3-30d: Samenvatting van de 80286 instructieset, deel 4.

3.3 80286

FUNCTION	FORMAT	CLOCK COUNT Real Address Mode	CLOCK COUNT Protected Virtual Address Mode	COMMENTS Real Address Mode	COMMENTS Protected Virtual Address Mode
CONTROL TRANSFER					
CALL = Call:					
Direct within segment	[11101000] [disp-low] [disp-high]	7+m	7+m	2	18
Register/memory indirect within segment	[11111111] [mod 010 r/m]	7+m, 11+m*	7+m, 11+m*	2,8	8,9,18
Direct intersegment	[10011010] [segment offset]	13+m	26+m	2	11,12,18
Protected Mode Only (Direct Intersegment):	[segment selector]				
Via call gate to same privilege level			41+m		8,11,12,18
Via call gate to different privilege level, no parameters			82+m		8,11,12,18
Via call gate to different privilege level, x parameters			86 +4x+m		8,11,12,18
Via TSS			177+m		8,11,12,18
Via task gate			182+m		8,11,12,18
Indirect intersegment	[11111111] [mod 011 r/m] (mod ≠ 11)	16+m	29+m*	2	8,9,11,12,18
Protected Mode Only (Indirect Intersegment):					
Via call gate to same privilege level			44+m*		8,9,11,12,18
Via call gate to different privilege level, no parameters			83 +m*		8,9,11,12,18
Via call gate to different privilege level, x parameters			90+4x +m*		8,9,11,12,18
Via TSS			180+m*		8,9,11,12,18
Via task gate			185+m*		8,9,11,12,18
JMP = Unconditional jump:					
Short/long	[11101011] [disp-low]	7+m	7+m		18
Direct within segment	[11101001] [disp-low] [disp-high]	7+m	7+ m		18
Register/memory indirect within segment	[11111111] [mod 100 r/m]	7 +m, 11+m*	7+m, 11+m*	2	9,18
Direct intersegment	[11101010] [segment offset]	11+m	23+m		11,12,18
Protected Mode Only (Direct Intersegment):	[segment selector]				
Via call gate to same privilege level			38+m		8,11,12,18
Via TSS			175+m		8,11,12,18
Via task gate			180+m		8,11,12,18
Indirect intersegment	[11111111] [mod 101 r/m] (mod ≠ 11)	15+m*	26+m*	2	8,9,11,12,18
Protected Mode Only (Indirect Intersegment):					
Via call gate to same privilege level			41+m*		8,9,11,12,18
Via TSS			178+m*		8,9,11,12,18
Via task gate			183+m*		8,9,11,12,18
RET = Return from CALL:					
Within segment	[11000011]	11+m	11+m	2	8,9,18
Within seg adding immed to SP	[11000010] [data-low] [data-high]	11+m	11+m	2	8,9,18
Intersegment	[11001011]	15+m	25+m	2	8,9,11,12,18
Intersegment adding immediate to SP	[11001010] [data-low] [data-high]	15+m		2	8,9,11,12,18
Protected Mode Only (RET):					
To different privilege level			55+m		9,11,12,18

Tabel 7/3.3-30e: Samenvatting van de 80286 instructieset, deel 5.

3.3 80286

FUNCTION	FORMAT	CLOCK COUNT Real Address Mode	CLOCK COUNT Protected Virtual Address Mode	COMMENTS Real Address Mode	COMMENTS Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
JE/JZ = Jump on equal zero	01110100 | disp	7 + m or 3	7 + m or 3		18
JL/JNGE = Jump on less/not greater or equal	01111100 | disp	7 + m or 3	7 + m or 3		18
JLE/JNG = Jump on less or equal/not greater	01111110 | disp	7 + m or 3	7 + m or 3		18
JB/JNAE = Jump on below/not above or equal	01110010 | disp	7 + m or 3	7 + m or 3		18
JBE/JNA = Jump on below or equal/not above	01110110 | disp	7 + m or 3	7 + m or 3		18
JP/JPE = Jump on parity/parity even	01111010 | disp	7 + m or 3	7 + m or 3		18
JO = Jump on overflow	01110000 | disp	7 + m or 3	7 + m or 3		18
JS = Jump on sign	01111000 | disp	7 + m or 3	7 + m or 3		18
JNE/JNZ = Jump on not equal/not zero	01110101 | disp	7 + m or 3	7 + m or 3		18
JNL/JGE = Jump on not less/greater or equal	01111101 | disp	7 + m or 3	7 + m or 3		18
JNLE/JG = Jump on not less or equal/greater	01111111 | disp	7 + m or 3	7 + m or 3		18
JNB/JAE = Jump on not below/above or equal	01110011 | disp	7 + m or 3	7 + m or 3		18
JNBE/JA = Jump on not below or equal/above	01110111 | disp	7 + m or 3	7 + m or 3		18
JNP/JPO = Jump on not par/par odd	01111011 | disp	7 + m or 3	7 + m or 3		18
JNO = Jump on not overflow	01110001 | disp	7 + m or 3	7 + m or 3		18
JNS = Jump on not sign	01111001 | disp	7 + m or 3	7 + m or 3		18
LOOP = Loop CX times	11100010 | disp	8 + m or 4	8 + m or 4		18
LOOPZ/LOOPE = Loop while zero/equal	11100001 | disp	8 + m or 4	8 + m or 4		18
LOOPNZ/LOOPNE = Loop while not zero/equal	11100000 | disp	8 + m or 4	8 + m or 4		18
JCXZ = Jump on CX zero	11100011 | disp	8 + m or 4	8 + m or 4		18
ENTER = Enter Procedure	11001000 | data-low | data-high | L			2,8	8,9
L = 0		11	11	2,8	8,9
L = 1		15	15	2,8	8,9
L > 1		16 + 4(L − 1)	16 + 4(L − 1)	2,8	8,9
LEAVE = Leave Procedure	11001001	5	5		
INT = Interrupt:					
Type specified	11001101 | type	23 + m		2,7,8	
Type 3	11001100	23 + m		2,7,8	
INTO = Interrupt on overflow	11001110	24 + m or 3 (3 if no interrupt)	(3 if no interrupt)	2,6,8	

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Tabel 7/3.3-30f: Samenvatting van de 80286 instructieset, deel 6.

3.3 80286

FUNCTION	FORMAT	CLOCK COUNT Real Address Mode	CLOCK COUNT Protected Virtual Address Mode	COMMENTS Real Address Mode	COMMENTS Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only:					
Via interrupt or trap gate to same privilege level			40 + m		7,8,11,12,18
Via interrupt or trap gate to fit different privilege level			78 + m		7,8,11,12,18
Via Task Gate			167 + m		7,8,11,12,18
IRET = Interrupt return	11001111	17 + m	31 + m	2,4	8,9,11,12,15,18
Protected Mode Only:					
To different privilege level			55 + m		8,9,11,12,15,18
To different task (NT = 1)			169 + m		8,9,11,12,18
BOUND = Detect value out of range	01100010 | mod reg r/m	13*	13* (Use INT clock count if exception 5)	2,6	6,8,9,11,12,18
PROCESSOR CONTROL					
CLC = Clear carry	11111000	2	2		
CMC = Complement carry	11110101	2	2		
STC = Set carry	11111001	2	2		
CLD = Clear direction	11111100	2	2		
STD = Set direction	11111101	2	2		
CLI = Clear interrupt	11111010	3	3		14
STI = Set interrupt	11111011	2	2		14
HLT = Halt	11110100	2	2		13
WAIT = Wait	10011011	3	3		
LOCK = Bus lock prefix	11110000	0	0		14
CTS = Clear task switched flag	00001111 | 00000110	2	2	3	13
ESC = Processor Extension Escape	11011TTT | mod LLL r/m (TTT LLL are opcode to processor extension)	9–20*	9–20*	5,8	8,17
SEG = Segment Override Prefix	001 reg 110	0	0		
PROTECTION CONTROL					
LGDT = Load global descriptor table register	00001111 | 00000001 | mod 010 r/m	11*	11*	2,3	9,13
SGDT = Store global descriptor table register	00001111 | 00000001 | mod 000 r/m	11*	11*	2,3	9
LIDT = Load interrupt descriptor table register	00001111 | 00000001 | mod 011 r/m	12*	12*	2,3	9,13
SIDT = Store interrupt descriptor table register	00001111 | 00000001 | mod 001 r/m	12*	12*	2,3	9
LLDT = Load local descriptor table register from register memory	00001111 | 00000000 | mod 010 r/m		17,19*	1	9,11,13
SLDT = Store local descriptor table register to register/memory	00001111 | 00000000 | mod 000 r/m		2,3*	1	9

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Tabel 7/3.3-30g: Samenvatting van de 80286 instructieset, deel 7.

3.3 80286

FUNCTION	FORMAT			CLOCK COUNT Real Address Mode	CLOCK COUNT Protected Virtual Address Mode	COMMENTS Real Address Mode	COMMENTS Protected Virtual Address Mode
PROTECTION CONTROL (Continued)							
LTR = Local task register from register/memory	00001111	00000000	mod 0 1 1 r/m		17,19*	1	9,11,13
STR = Store task register to register memory	00001111	00000000	mod 0 0 1 r/m		2,3*	1	9
LMSW = Load machine status word from register/memory	00001111	00000001	mod 1 1 0 r/m	3,6*	3,6*	2,3	9,13
SMSW = Store machine status word	00001111	00000001	mod 1 0 0 r/m	2,3*	2,3*	2,3	9
LAR = Load access rights from register/memory	00001111	00000010	mod reg r/m		14,16*	1	9,11,16
LSL = Load segment limit from register/memory	00001111	00000011	mod reg r/m		14,16*	1	9,11,16
ARPL = Adjust requested privilege level: from register/memory		01100011	mod reg r/m		10*,11*	2	8,9
VERR = Verify read access: register/memory	00001111	00000000	mod 1 0 0 r/m		14,16*	1	9,11,16
VERR = Verify write access:	00001111	00000000	mod 1 0 1 r/m		14,16*	1	9,11,16

Shaded areas indicate instructions not available in 8086, 88 microsystems.

Tabel 7/3.3-30h: Samenvatting van de 80286 instructieset, deel 8.

3.3 80286

Footnotes

The Effective Address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EQ = disp-high: disp-low.

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

reg is assigned according to the following:

reg	Segment Register
00	ES
01	CS
10	SS
11	DC

REG is assigned according to the following table:

16-Bit (w = 1)		8-Bit (w = 0)	
000	AX	000	AL
001	CX	001	CL
010	DX	010	DL
011	BX	011	BL
100	SP	100	AH
101	BP	101	CH
101	SI	110	DH
111	DI	111	BH

The physical addresses of all operands addressed by the BP register are computed using the SS segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

Tabel 7/3.3-31: Toelichting op de 80286 instructieset.

7/3.4

80C286

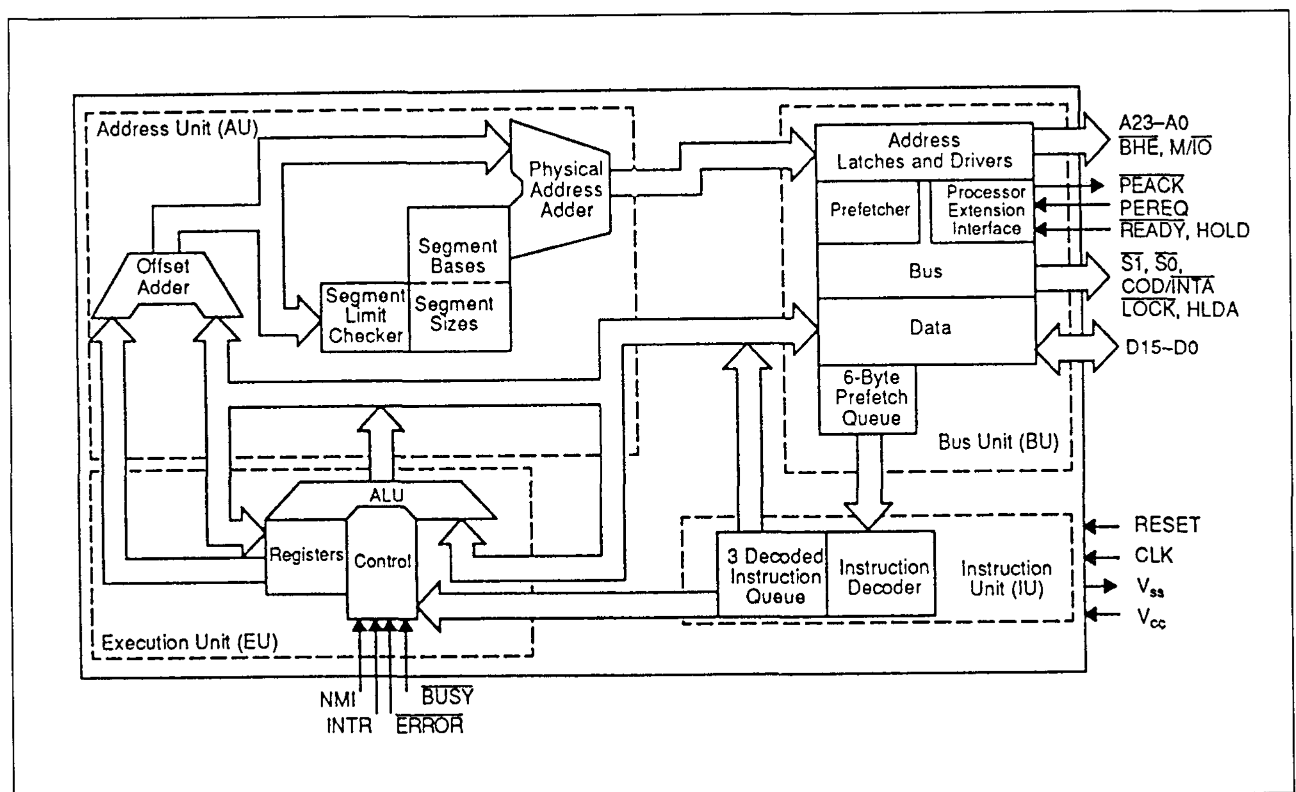
Algemeen

Beschrijving

De 80C286 is een high-speed CMOS-versie van de industriestandaard 80286 microprocessor.

De 80C286 is functioneel en wat aansluitingen betreft 100 % identiek met de NMOS-versie (plug-in vervanger). In dit gedeelte worden dan ook alleen de optredende signalen behandeld en de gegevens die afwijken van het NMOS-type 80286.

Door de toegepaste CMOS processen zijn hogere kloksnelheden mogelijk (tot 25 MHz), terwijl ook bij lagere frequenties (tot en met DC) volledige registerstatus behouden blijft. Dit laatste is belangrijk als er energie bespaard moet worden. In de standby-mode wordt bijvoorbeeld slechts 5 mA uit de voeding opgenomen.

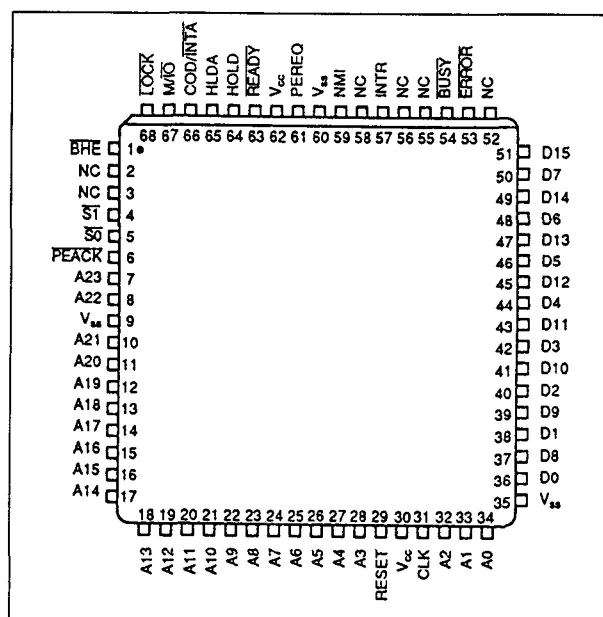


Figuur 7/3.4-1: Blokschema van de 80C286 microprocessor.

3.4 80C286

Algemene gegevens

- 100 % compatibel met NMOS-versie 80286
- clockfrequenties:
  12,5 MHz, 16 MHz, 20 MHz
  (respectievelijk 80C286-12, -16 en -20)
- statisch CMOS-ontwerp voor geringe dissipatie
- voedingsstroom:
  standby: 5 mA
  12,5 MHz: 220 mA
  16 MHz: 260 mA
  20 MHz: 310 mA
- behuizing: 68-pens plastic leaded chip-carrier (PLCC)
- Fabrikanten en versies:
  AMD: 80C286 (-12, -16, -20)
  Harris: 80C286 (-12, -16, -20, -25)
  Intel: 80C286 (12,5 MHz)



Figuur 7/3.4-2: Bovenaanzicht van de aansluitingen van de 80C286.

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	BHE	24	A7	47	D13
2	NC	25	A6	48	D6
3	NC	26	A5	49	D14
4	S1	27	A4	50	D7
5	S0	28	A3	51	D15
6	PEACK	29	RESET	52	NC
7	A23	30	V_{cc}	53	ERROR
8	A22	31	CLK	54	BUSY
9	V_{ss}	32	A2	55	NC
10	A21	33	A1	56	NC
11	A20	34	A0	57	INTR
12	A19	35	V_{ss}	58	NC
13	A18	36	D0	59	NMI
14	A17	37	D8	60	V_{ss}
15	A16	38	D1	61	PEREQ
16	A15	39	D9	62	V_{cc}
17	A14	40	D2	63	READY
18	A13	41	D10	64	HOLD
19	A12	42	D3	65	HLDA
20	A11	43	D11	66	COD/INTA
21	A10	44	D4	67	M/IO
22	A9	45	D12	68	LOCK
23	A8	46	D5		

Tabel 7/3.4-1: Penfuncties van de 80C286.

3.4 80C286

Aansluitingen en pen-functies

A23 tot en met A0
uitgangen (aktief-HOOG)
Dit zijn ADDRESS BUS uitgangssignalen waarmee fysiek geheugen en I/O-poorten worden geadresseerd. A0 is LAAG wanneer data moet worden overgebracht via de pennen D7 tot en met D0. A23 tot en met A16 zijn LAAG tijdens I/O-transfers. De adresbus is aktief-HOOG en zwevend bij bus hold-acknowledge.

$\overline{BHE}$
uitgang (aktief-LAAG)
Het BUS HIGH ENABLE-signaal geeft aan dat data-transport op de hoogste byte van de databus (D15 tot en met D8) plaats vindt. 8 bit schakelingen die op de hoogste byte van de databus zijn aangesloten gebruiken $\overline{BHE}$ meestal voor chip-select functies. $\overline{BHE}$ is aktief-LAAG en hoog-impedant tijdens bus hold acknowledge.

$\overline{BHE}$ Value	A0 Value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus (D15–D8)
1	0	Byte transfer on lower half of data bus (D7–D0)
1	1	Reserved

Tabel 7/3.4-2: Coderingen van $\overline{BHE}$ en A0.

$\overline{BUSY}$, $\overline{ERROR}$
ingangen (aktief-LAAG)
PROCESSOR EXTENSION BUSY- en ERROR geven de 80C286 informatie over de bedrijfsconditie van een processor-uitbreiding. Een aktief $\overline{BUSY}$-signaal maakt dat de 80C286 stopt met de programma-uitvoering op WAIT en sommige ESC-instructies totdat $\overline{BUSY}$ weer niet-aktief (HOOG) wordt. De 80C286 kan al wachtend op het niet-aktief worden van $\overline{BUSY}$ worden geïnterrumpeerd. Een aktief $\overline{ERROR}$-signaal maakt dat de 80C286 een processor-uitbreidingsinterrupt geeft bij het uitvoeren van WAIT of bepaalde ESC-instructies. Deze ingangen zijn aktief-LAAG en mogen asynchroon ten opzichte van de systeemclock werken.

CLK
ingang (aktief-HOOG)
SYSTEM CLOCK voert de fundamentele timing van 80C286 systemen uit. Het wordt binnen de 80C286 door twee gedeeld voor het opwekken van de processor-clock. De interne deel-door-twee schakeling kan op een externe clockgenerator worden gesynchroniseerd door een LAAG-naar-HOOG overgang op de RESET-ingang.

COD/$\overline{INTA}$
uitgang
Met CODE/INTERRUPT ACKNOWLEDGE wordt verschil gemaakt tussen instructie ophaal-cycli en geheugen data-leescycli. Tevens worden interrupt-acknowledge cycli onderscheiden van I/O-cycli. De COD/$\overline{INTA}$-uitgang wordt intern opgetrokken gedurende bus hold-acknowledge.

D15 tot en met D0
in-/uitgangen (aktief-HOOG)
Tijdens leescycli voor geheugen-, I/O- en interrupt-acknowledge wordt data via de DATABUS opgenomen, terwijl data (door de 80C286) wordt afgegeven tijdens schrijfcycli voor geheugen- en I/O-acties. De databus is aktief-HOOG en is zwevend (3-state "uit") gedurende bus hold-acknowledge.

HOLD, HLDA
ingang/uitgang (aktief-HOOG)
BUS HOLD REQUEST en HOLD ACKNOWLEDGE besturen het bezit van de lokale 80C286-bus. Met het HOLD-signaal kan een andere lokale busmaster om controle van de lokale bus vragen. Als dit is toegestaan maakt de 80C286 zijn busdrivers zwevend (3-state "uit") en aktiveert dan HLDA, waardoor de bus hold-acknowledge conditie

wordt bereikt. De lokale bus blijft aan de vragende master toegewezen totdat HOLD niet-aktief wordt. Hierdoor wordt HLDA gedeactiveerd en neemt de 80C286 de besturing van de lokale bus weer over waardoor de bus hold-acknowledge conditie wordt beëindigd. HOLD kan asynchroon zijn ten opzichte van de systeemclock. Beide signalen zijn aktief-HOOG.

INTR
ingang (aktief-HOOG)
Het INTERRUPT REQUEST-signaal verzoekt de 80C286 om de uitvoering van het lopende programma uit te stellen en eerst service te verlenen aan een extern verzoek. Interrupt-requests worden gemaskeerd als het interrupt-enable bit in het flag-woord is gecleared. Wanneer de 80C286 op een interrupt-request reageert, voert hij twee interrupt-acknowledge buscycli uit om een 8 bit interrupt-vector in te lezen die de herkomst van de interruptie aangeeft. Om te garanderen dat het programma wordt onderbroken moet INTR aktief blijven totdat de eerste interrupt-acknowledge cyclus is volbracht. INTR wordt aan het begin van elke processorcyclus afgetast en moet tenminste twee processorcycli vóór beëindiging van de lopende instructie aktief-HOOG zijn om vóór de volgende instructie te kunnen interrumperen. INTR is niveau-gevoelig, aktief-HOOG en kan asynchroon met de systeemclock optreden.

$\overline{\text{LOCK}}$
uitgang (aktief-LAAG)
BUS LOCK geeft aan dat andere systeembusmasters na de lopende buscyclus geen controle over de systeembus mogen krijgen. Het $\overline{\text{LOCK}}$-signaal kan expliciet worden geaktiveerd door de LOCK instructie-prefix of automatisch door 80C286 hardware tijdens geheugen XCHG-instructies, interrupt-acknowledge of toegang tot de sleutelwoordentabel. $\overline{\text{LOCK}}$ is aktief-LAAG en zwevend (3-state "uit") gedurende bus-hold acknowledge.

M/$\overline{\text{IO}}$
uitgang
MEMORY/IO SELECT maakt onderscheid tussen geheugen-toegang en I/O-toegang. Als dit signaal tijdens T_S HOOG is, is een geheugen-cyclus of een halt/shutdown-cyclus bezig. Is het signaal LAAG, dan loopt een I/O-cyclus of een interrupt-acknowledge cyclus. M/$\overline{\text{IO}}$ is zwevend (3-state "uit") gedurende bus-hold acknowledge.

NC
no connect
No Connect-pennen mogen nergens mee verbonden worden!

NMI
ingang (aktief-HOOG)
Het NON-MASKABLE INTERRUPT REQUEST-signaal interrumpeert de 80C286 met een intern geleverde vectorwaarde van 2. Er worden geen interrupt-acknowledge cycli uitgevoerd. De interrupt-enable bit in het flag-woord van de 80C286 heeft geen invloed op deze ingang. Het NMI-signaal is aktief-HOOG, mag asynchroon zijn ten opzichte van de systeemclock en wordt na interne synchronisatie flank-getriggerd. Om goed herkend te worden moet het signaal eerst tenminste vier cycli van de systeemclock LAAG zijn geweest, waarna het zeker vier systeem clockcycli HOOG moet blijven.

PEREQ, $\overline{\text{PEACK}}$
ingang/uitgang (aktief-HOOG)
De PROCESSOR EXTENSION OPERAND REQUEST-ingang en de PROCESSOR EXTENSION ACKNOWLEDGE-uitgang geven de 80C286 mogelijkheden van geheugenmanagement en beveiliging naar processoruitbreidingen. Het PEREQ-signaal vraagt de 80C286 een data operand-transfer uit te voeren voor een processor-uitbreiding.
De $\overline{\text{PEACK}}$-uitgang laat de uitbreiding weten wanneer de gevraagde operand wordt overgebracht. PEREQ is aktief-HOOG en mag asynchroon zijn ten opzichte van de systeemclock. $\overline{\text{PEACK}}$ is aktief-LAAG.

3.4 80C286

$\overline{\text{READY}}$
ingang (aktief-LAAG)

Het BUS READY-signaal beëindigt een buscyclus. Buscycli worden onbegrensd verlengd totdat zij worden beëindigd door het LAAG gaan van $\overline{\text{READY}}$. Bus Ready heeft aan de systeemclock gerefereerde setup- en houdtijden nodig om correct te werken.
$\overline{\text{READY}}$ wordt gedurende bus hold-acknowledge genegeerd.

RESET
ingang (aktief-HOOG)

De interne logika van de 80C286 wordt met SYSTEM RESET leeg gemaakt. SYSTEM RESET is aktief-HOOG. De 80C286 kan op elk willekeurig moment opnieuw worden geïnitialiseerd door een LAAG-naar-HOOG overgang van het RESET-signaal wanneer dat langer dan 16 systeem-clockcycli aktief blijft. Wanneer RESET aktief is gaan de uitgangen van de 80C286 in de aangegeven toestanden, zie tabel 7/3.4-3.

Pin Value	Pin Names
1 (High)	$\overline{\text{S0}}$, $\overline{\text{S1}}$, $\overline{\text{PEACK}}$, A23–A0, $\overline{\text{BHE}}$, $\overline{\text{LOCK}}$
0 (Low)	M/$\overline{\text{IO}}$, COD/$\overline{\text{INTA}}$, HLDA
Three-state Off	D15–D0

Tabel 7/3.4-3: Toestanden van de pennen tijdens het resetten van de 80C286.

De 80C286 begint te werken na een HOOG-naar-LAAG overgang op RESET. Deze overgang moet synchroon zijn met de systeemclock. De 80C286 heeft ongeveer 50 systeem-clockcycli nodig voor interne initialisatie voordat de eerste buscyclus wordt uitgevoerd (ophalen van code uit het power-on executie-adres).
Door een synchroon met de systeemclock optredende LAAG-naar-HOOG overgang van RESET wordt een nieuwe processorcyclus gestart op de volgende HOOG-naar-LAAG overgang van de systeemclock. De LAAG-naar-HOOG overgang van RESET mag asynchroon met de systeemclock optreden, maar dan kan niet worden voorspeld welke fase van de processorclock zal optreden gedurende de volgende systeem-periode. Synchrone LAAG-naar-HOOG overgangen zijn alleen vereist in systemen waarbij de processorclock fase-synchroon moet zijn met een andere clock.

$\overline{\text{S1}}$, $\overline{\text{S0}}$
uitgangen (aktief-LAAG)

Met BUS CYCLE STATUS wordt het initialiseren van een buscyclus aangegeven en wordt samen met M/$\overline{\text{IO}}$ en COD/$\overline{\text{INTA}}$ het type van de buscyclus bepaald. De bus is zwevend als $\overline{\text{S1}}$ en/of $\overline{\text{S2}}$ LAAG is. $\overline{\text{S1}}$ en $\overline{\text{S2}}$ zijn aktief-LAAG en worden tijdens bus hold-acknowledge intern opgetrokken.

COD/ $\overline{\text{INTA}}$	M/$\overline{\text{IO}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$	Bus cycle initiated
0 (Low)	0	0	0	Interrupt acknowledge
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	None; not a status cycle
0	1	0	0	If A_1 = 1 then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (High)	0	0	0	Reserved
1	0	0	1	I/O Read
1	0	1	0	I/O Write
1	0	1	1	None; not a status cycle
1	1	0	0	Reserved
1	1	0	1	Memory instruction read
1	1	1	0	Reserved
1	1	1	1	None; not a status cycle

Tabel 7/3.4-4: Definities van de 80C286 buscyclus-status.

V_{SS} (GND)
ingang

De GROUND-ingang wordt verbonden met de systeem-massa (aarde: 0 V).

V_{CC}
(ingang)

De SYSTEM POWER-ingang wordt aangesloten op de +5 V systeemvoeding, tolerantie +/-5 %.

3.4 80C286

Overige kenmerken

Voor de interne architectuur, registerset, algemene registers, segment-, basis- index-, status- en controlregisters wordt verwezen naar de beschrijving van de 80286 microprocessor. Ook de instructieset en geheugenorganisatie zijn identiek voor beide typen. In de tabellen 7/3.4-5 tot en met -9 en de figuren 7/3.4-3 tot en met -8 zijn de afwijkende elektrische en timing-karakteristieken van de 80C286 vermeld.

Supply Voltage	+8.0 V
Input, Output or I/O	
Voltage Applied GND	−1.0 V to V_{DD}+1.0 V
Power Dissipation/Speed	
20 MHz	1.7 W
16 MHz	1.4 W
12.5 MHz	1.2 W
Storage Temperature Range	−65°C to +150°C
Junction Temperature	+165°C
Lead Temperature	
(Soldering, Ten Seconds)	+275°C

Tabel 7/3.4-5: Maximaal toegelaten waarden.

Operating Voltage Range	+4.5 V to +5.5 V
80C286-20 Only	+4.75 V to +5.25 V
Operating Temperature	
Range	0 to +100°C case temperature
(Meets laptop temperature requirements.)	

Tabel 7/3.4-6: Aanbevolen bedrijfscondities van de 80C286.

Parameter Symbol	Parameter Description	Typ	Unit	Test Conditions
C_{CLK}	CLK Input Capacitance	10	pF	FREQ = 1 MHz
C_{IN}	Other Input Capacitance	10	pF	
$C_{I/O}$	I/O Capacitance	10	pF	

Tabel 7/3.4-7: Capaciteiten bij 1 MHz.

3.4 80C286

V_{CC} = +5 V ± 10% for 80C286-12 and 80C286-16; V_{CC} = +5 V ±5% for 80C286-20, T_C = 0°C to +100°C

Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
V_{IL}	Input Low Voltage		-0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.5	V
V_{ILC}	CLK Input Low Voltage		-0.5	1.0	V
V_{IHC}	CLK Input High Voltage		3.6	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	I_{OL} = 2.0 mA	–	0.4	V
V_{OH}	Output High Voltage	I_{OH} = –2.0 mA I_{OH} = –100 mA	3.0 V_{CC} – 0.4	–	V V
I_I	Input Leakage Current	V_{IN} = GND or V_{CC} Pins 29, 31, 57, 59, 61, 63–64	–10	10	µA
I_{SH}	Input Sustaining Current on $\overline{BUSY}$ and $\overline{ERROR}$ Pins	V_{IN} = GND (see Note 5)	–30	–500	µA
I_{BHL}	Input Sustaining Current High	V_{IN} = 1.0 V (see Note 1)	38	200	µA
I_{BHH}	Input Sustaining Current High	V_{IN} = 3.0 V (see Note 2)	–50	–400	µA
I_O	Output Leakage Current	V_O = GND or V_{CC} Pins 1, 7–8, 10–28, 32–34	–10	10	µA
I_{CCOP}	Active Power Supply Current	80C286-12 (see Note 4) 80C286-16 (see Note 4) 80C286-20 (see Note 4)	– – –	220 260 310	mA mA mA
I_{CCSB}	Standby Power Supply Current	(see Note 3)	–	5	mA

Tabel 7/3.4-8: Gelijkspanningskarakteristieken van de 80C286.

3.4 80C286

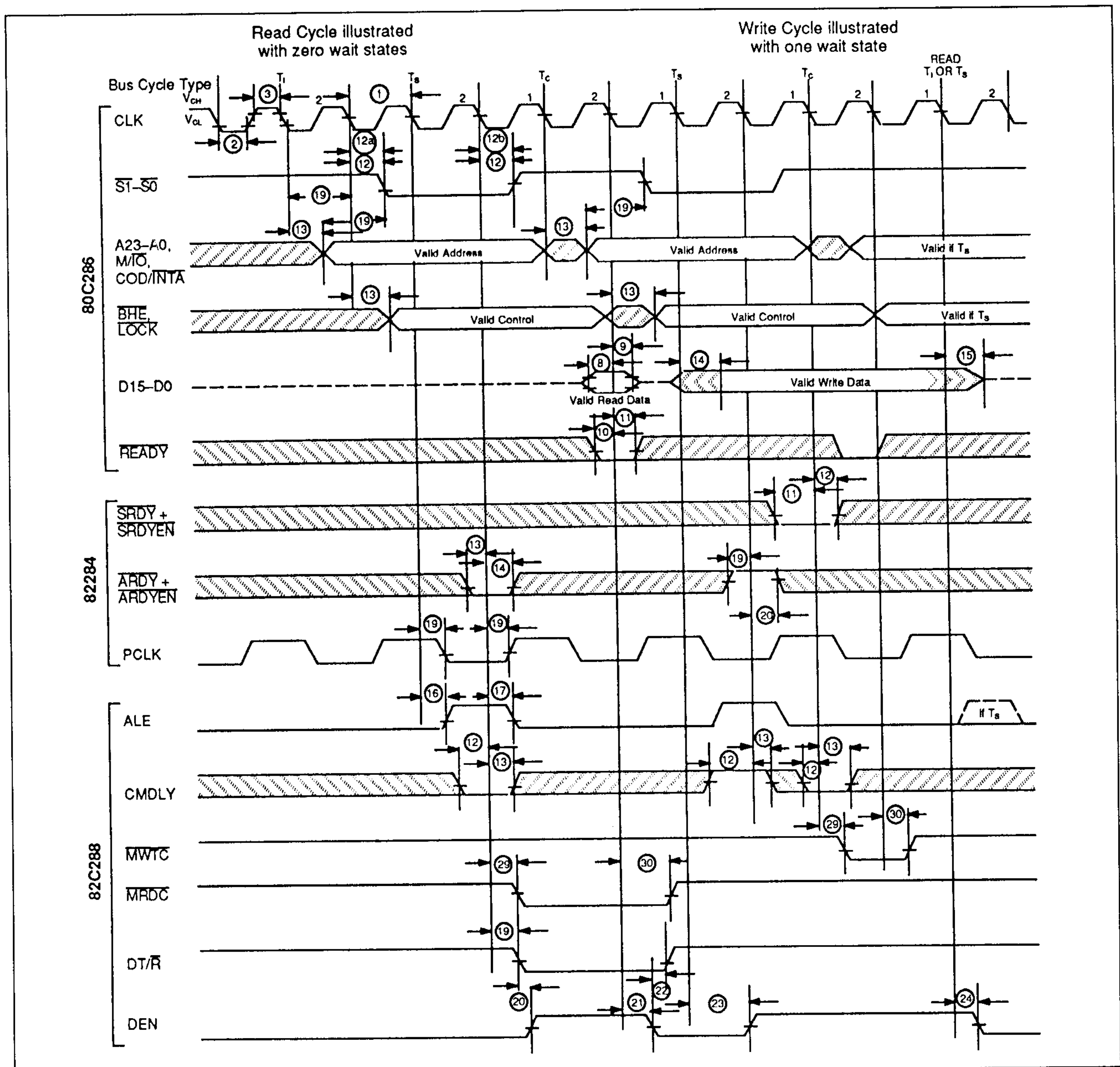
Vcc = +5 V ± 10% for 80C286-12 and 80C286-16; Vcc = +5 V ±5% for 80C286-20, Tc = 0°C to +100°C
AC Timings are referenced to 0.8 V and 2.0 V points of the signals as illustrated in datasheet waveforms, for 12.5 and 16 MHz, unless otherwise specified. For 20 MHz, AC timings are referenced to the 1.5 V point of the signals as illustrated in Data Sheet waveforms, unless otherwise specified.

Parameter		Test	12.5 MHz		16 MHz		20 MHz		
Symbol	Description	Conditions	Min	Max	Min	Max	Min	Max	Units
Timing Requirements									
1	System Clock (CLK) Period		40	–	31	–	25	–	ns
2	System Clock (CLK) Low Time	@ 1.0 V	11	–	7	–	6	–	ns
3	System Clock (CLK) High Time	@ 3.6 V	13	–	11	–	9	–	ns
17	System Clock (CLK) RISE Time	1.0 V to 3.6 V	–	8	–	5	–	4	ns
18	System Clock (CLK) FALL Time	3.6 V to 1.0 V	–	8	–	5	–	4	ns
4	Asynchronous Inputs SETUP Time	(Note 1)	15	–	5	–	4	–	ns
5	Asynchronous Inputs HOLD Time	(Note 1)	15	–	5	–	4	–	ns
6	RESET SETUP Time		10	–	10	–	10	–	ns
7	RESET HOLD Time		0	–	0	–	0	–	ns
8	Read Data SETUP Time		5	–	5	–	3	–	ns
9	Read Data HOLD Time		4	–	3	–	2	–	ns
10	$\overline{READY}$ SETUP Time		20	–	12	–	10	–	ns
11	$\overline{READY}$ HOLD Time		20	–	5	–	3	–	ns
20	Input RISE/FALL Times	0.8 V to 2.0 V	–	8	–	6	–	6	ns
Timing Responses									
12A	Status/$\overline{PEACK}$ Active Delay	1, (Notes 3, 6, 7)	1	21	1	18	1	15	ns
12B	Status/$\overline{PEACK}$ Inactive Delay	1, (Notes 3, 6)	1	24	1	20	1	16	ns
13	Address Valid Delay	1, (Notes 2, 3)	1	32	1	27	1	23	ns
14	Write Data Valid Delay	1, (Notes 2, 3)	0	31	0	28	0	27	ns
15	Address/Status/Data Float Delay	2, (Note 5)	0	32	0	29	0	25	ns
16	HLDA Valid Delay	1, (Notes 2, 3, 8)	0	25	0	25	0	20	ns
19	Address Valid–Status SETUP Time	1, (Notes 3, 4)	22	–	16	–	9	–	ns

Notes:
1. Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{ERROR}$, and $\overline{BUSY}$. This specification is given only for testing purposes to assure recognition at a specific CLK edge.
2. Delay from 1.0 V on the CLK to 0.8 V or 2.0 V.
3. Output load: $C_L = 100$ pF.
4. Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 0.8 V or status going inactive reaching 2.0 V.
5. Delay from 1.0 V on the CLK to Float (no current drive) condition.
6. Delay from 1.0 V on the CLK to 0.8 V for Min (HOLD time) and to 2.0 V for Max (inactive delay).
7. Delay from 1.0 V on the CLK to 2.0 V for Min (HOLD time) and to 0.8 V for Max (active delay).
8. Delay from 1.0 V on the CLK to 2.0 V.

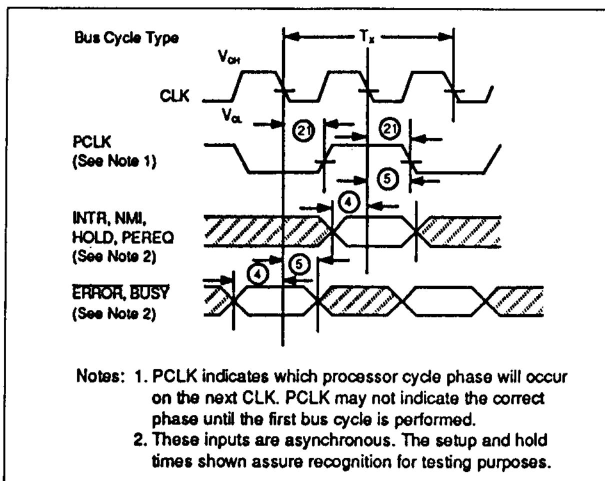
Tabel 7/3.4-9: Schakeltijden van de 80C286 bij verschillende clock-frequenties.

3.4 80C286

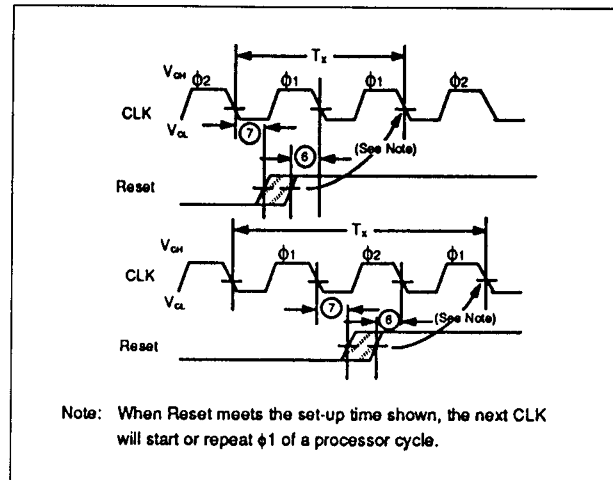


Figuur 7/3.4-3: De belangrijkste schakeltijden van de 80C286, de clockdriver 82284 en de buscontroller 82C288 (zie ook tabel 7/3.4-9).

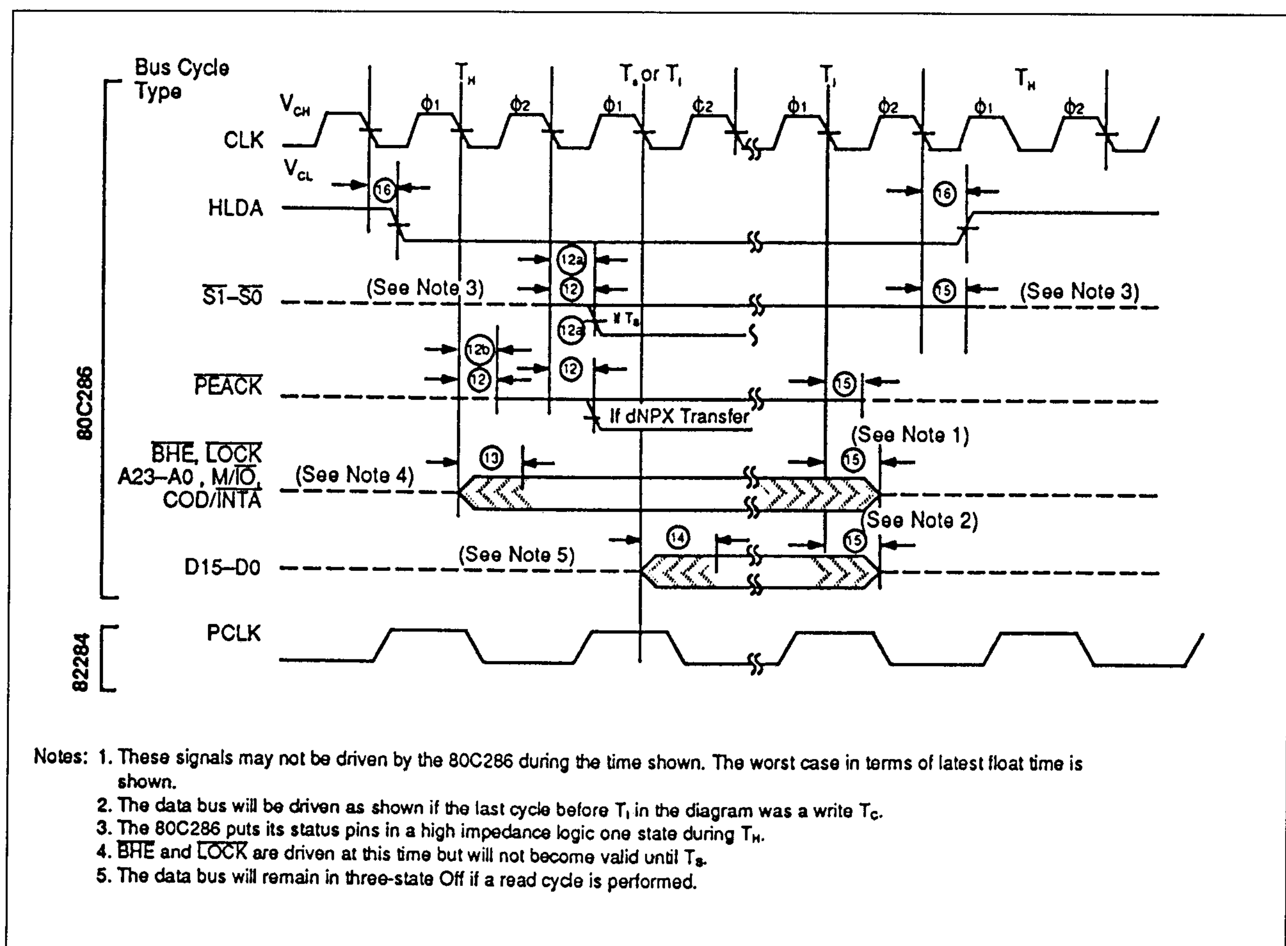
3.4 80C286



Figuur 7/3.4-4: Timing van asynchrone ingangssignalen.

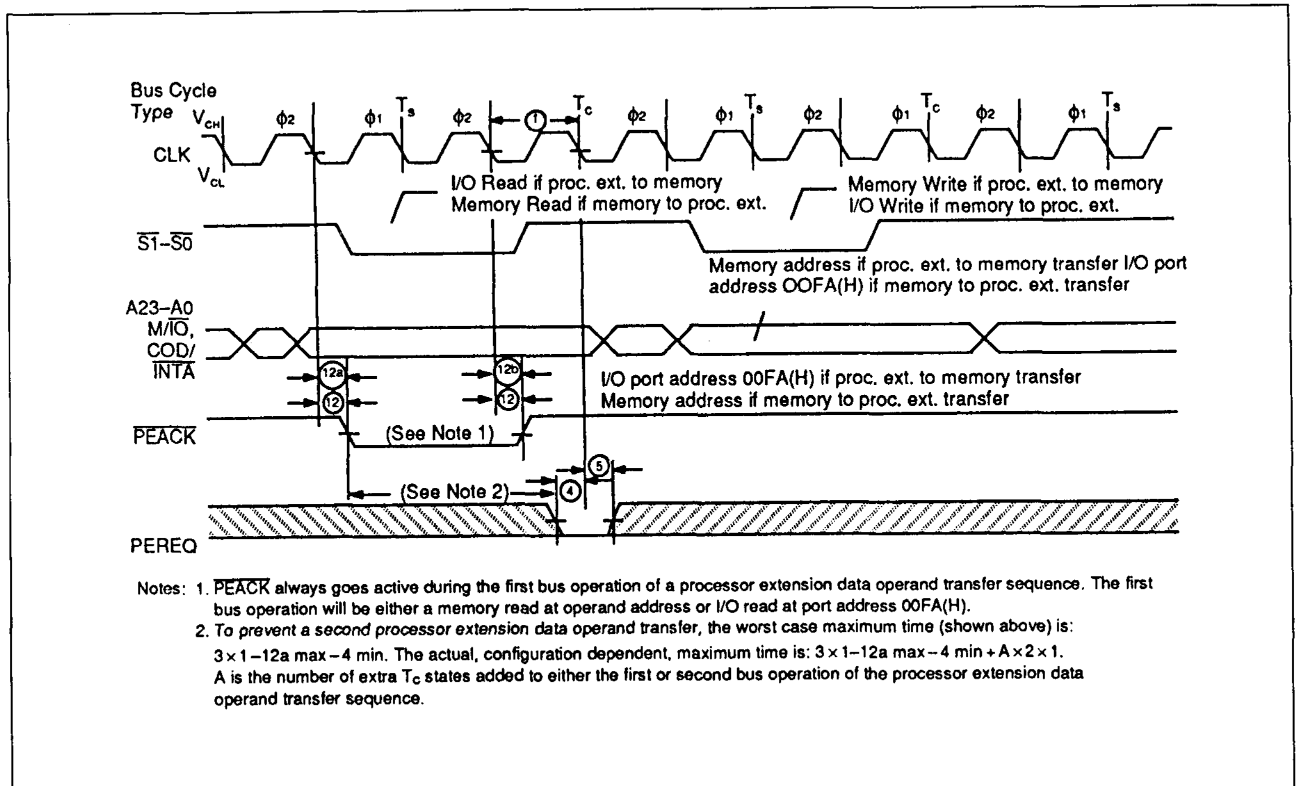


Figuur 7/3.4-5: Timing van reset en fase van de volgende processorcyclus.



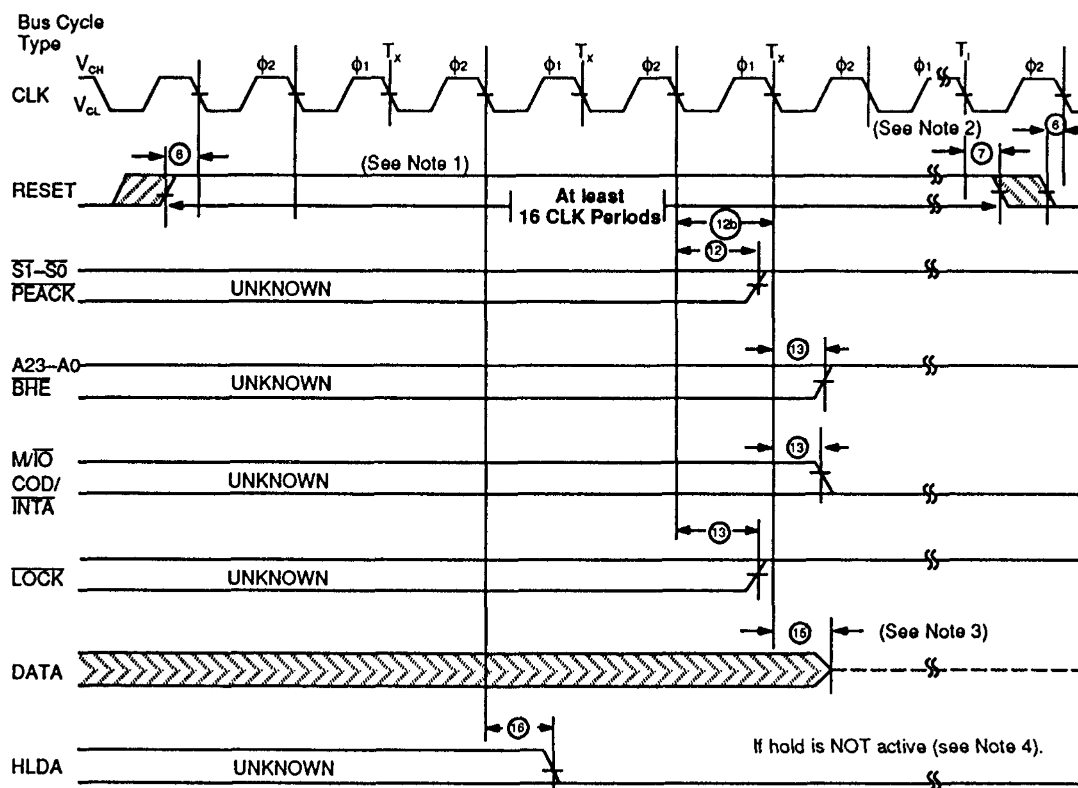
Figuur 7/3.4-6: Het oproepen van de hold-situatie.

3.4 80C286



Figuur 7/3.4-7: Timing van PEREQ/PEACK.

3.4 80C286



Notes: 1. Set-up time for RESET ↑ may be violated with the consideration that $\phi 1$ of the processor clock may begin one system CLK period later.
2. Set-up and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during $\phi 1$ or $\phi 2$.
3. The data bus is only guaranteed to be in three-state Off at the time shown.
4. HOLD is acknowledged during RESET, causing HLDA to go active and the appropriate pins to float. If HOLD remains active while RESET goes inactive, the 80C286 remains in HOLD state and will not perform any bus accesses until HOLD is deactivated.

Figuur 7/3.4-8: Initiële toestand van de signalen van de 80C286 tijdens het resetten.

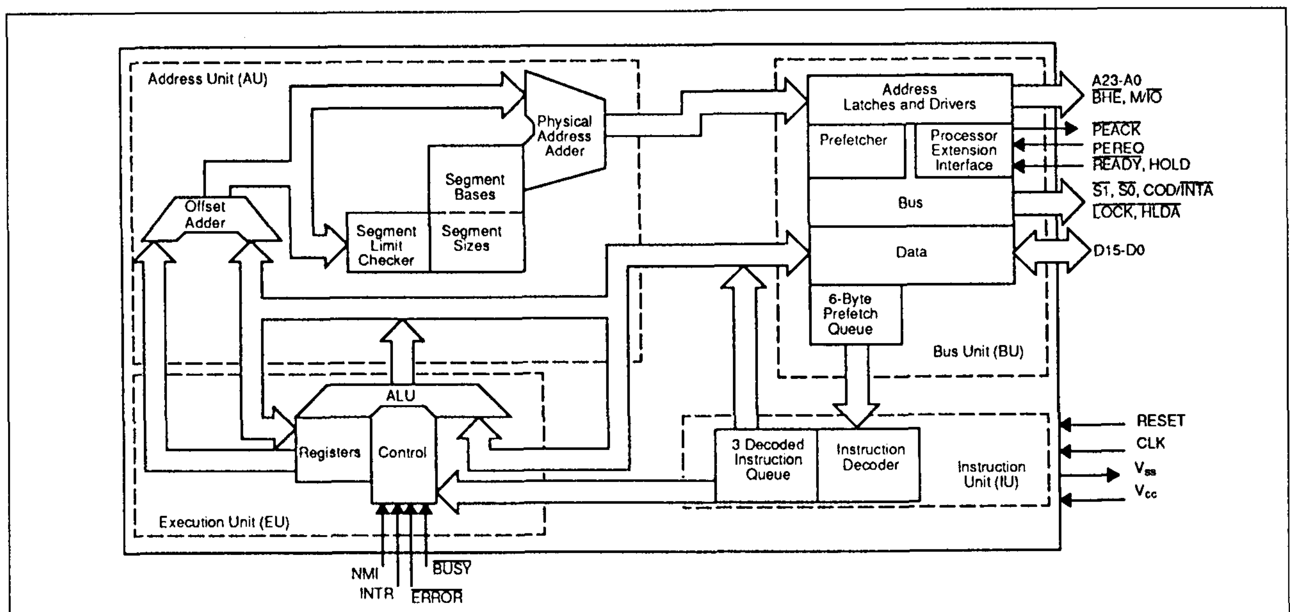
7/3.5

80L286

Algemeen

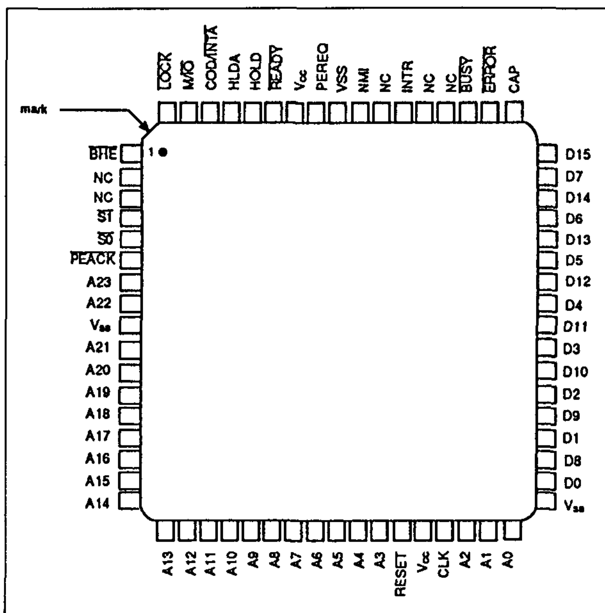
Beschrijving

De 80L286 is een geavanceerde high-performance microprocessor die, op het lagere energieverbruik na, identiek is aan de 80286. Door de verminderde dissipatie kan de 80L286 in een plastic leaded chip-carrier (PLCC) worden opgenomen, zonder dat daarbij een koellichaam nodig is. De PLCC-behuizing is niet alleen geschikt voor oppervlakte-montage maar ook om in een socket gestoken te worden. De "footprint" van de PLCC, LCC of PGA-behuizingen zijn gelijk, zodat de layout van de print niet veranderd behoeft te worden. De 80L286 is verkrijgbaar in 8, 10, 12 en 16 MHz uitvoeringen en is volledig compatibel met de 82C288 Bus Controller en de 82284 Clock Driver. De 80L286 is opwaarts compatibel met iAPX86 en -88 software. Wanneer de iAPX reële adresseermode wordt gebruikt is de 80L286 object-code compatibel met bestaande iAPX86 en -88 software. In de beveiligde virtuele adresseermode is de 80L286 hiermee source-code compatibel en kan upgrading nodig zijn om de beveiligingsmechanismen en virtuele adressen te kunnen gebruiken die door het geïntegreerde geheugen-management van de 80L286 worden ondersteund. In beide modes komen de eigenschappen van de 80L286 volledig tot hun recht en wordt een superset van de iAPX86 en -88 software uitgevoerd.



Figuur 7/3.5-1: Blokschema van de 80L286 microprocessor.

3.5 80L286



Figuur 7/3.5-2: Bovenaanzicht van de PL068-behuizing en aansluitingen van de 80L286.

De 80L286 voorziet in speciale operaties voor een efficiënte implementatie van bedrijfssystemen. Eén instructie kan bijvoorbeeld de uitvoering van een taak beëindigen, de status hiervan opbergen, overschakelen op een nieuwe taak, de status daarvan laden en de uitvoering van de nieuwe taak starten. De 80L286 ondersteunt ook virtuele geheugen-systemen door middel van een segment-not-ready uitzondering en herstartbare instructies.

Algemene gegevens

- identiek aan de NMOS 80286, maar dissipeert minder
- clockfrequenties:
  8, 10, 12,5 en 16 MHz
  (respectievelijk 80L286-8, -10, 12 en -16)
- behuizing: 68-pens plastic leaded chipcarrier (PLCC)
- socketed PLCC footprint gelijk aan socketed LCC en PGA
- oppervlakte-monteerbare PLCC
- grote adresseerruimte:
  16 MB fysiek,
  1 GB virtueel geheugen per taak
- geïntegreerd geheugen-management
- geheugenbescherming op 4 niveaus
- ondersteuning van virtueel geheugen en bedrijfssystemen
- Fabrikant: AMD 80L286 (-8, -10, -12, -16)

Beschrijving van de pen-functies

CLK
ingang (aktief-HOOG)

SYSTEM CLOCK levert de fundamentele timing van 80L286 systemen. Het is een 16 MHz signaal dat in de 80L286 door twee wordt gedeeld om de 8 MHz processor-clock te genereren. Het interne deel-door-twee circuit kan met een LAAG-naar-HOOG overgang op de RESET-ingang op een externe clockgenerator worden gesynchroniseerd.

D0 tot en met D15
in-/uitgangen (aktief-HOOG)

Data via de DATABUS opgenomen gedurende leescycli voor geheugen-, I/O- en interrupt-acknowledge, terwijl data wordt afgestaan tijdens schrijfcycli voor geheugen- en I/O. De databus is aktief-HOOG en is 3-state "uit" gedurende bus hold-acknowledge.

A23 tot en met A0
uitgangen (aktief-HOOG)

De ADDRESS BUS geeft uitgangssignalen af voor de adressering van fysiek geheugen en I/O-poorten. A0 is LAAG als data moet worden overgebracht via de pennen D7 tot en met D0. A23 tot en met A16 zijn LAAG tijdens I/O-transfers. De adresbus is aktief-HOOG en is 3-state "uit" bij bus hold-acknowledge.

$\overline{\text{BHE}}$
uitgang (aktief-LAAG)

BUS HIGH ENABLE geeft aan dat datatransport op de hoogste byte van de databus (D15 tot en met D8) plaats vindt. Gewoonlijk gebruiken 8 bit schakelingen die op de hoog-

3.5 80L286

ste byte van de databus zijn aangesloten $\overline{\text{BHE}}$ voor chip-select functies. $\overline{\text{BHE}}$ is aktief-LAAG en zweeft tijdens bus hold acknowledge.

$\overline{\text{BHE}}$ and A_0 Encodings

$\overline{\text{BHE}}$ Value	A0 Value	Function
0	0	Word transfer
0	1	Byte transfer on upper half of data bus (D15–D8)
1	0	Byte transfer on lower half of data bus (D7–D0)
1	1	Reserved

Tabel 7/3.5-1: *Coderingen van $\overline{\text{BHE}}$ en A0.*

$\overline{\text{S1}}$, $\overline{\text{S0}}$

uitgangen (aktief-LAAG)

Met de BUS CYCLE STATUS-signalen wordt het initialiseren van een buscyclus aangegeven en wordt, samen met M/$\overline{\text{IO}}$ en COD/$\overline{\text{INTA}}$, het type van de buscyclus bepaald. De bus is zwevend als één van beide LAAG is. $\overline{\text{S1}}$ en $\overline{\text{S2}}$ zijn aktief-LAAG en zweven tijdens bus hold-acknowledge.

M/$\overline{\text{IO}}$

uitgang

Met MEMORY/IO SELECT wordt onderscheid gemaakt tussen geheugen-toegang en I/O-toegang. Als dit signaal tijdens T_S HOOG is, is een geheugen-cyclus of een halt/shutdown-cyclus aan de gang. Is het signaal LAAG, dan is een I/O-cyclus of een interrupt-acknowledge cyclus bezig.
M/$\overline{\text{IO}}$ is zwevend gedurende bus-hold acknowledge.

COD/$\overline{\text{INTA}}$

uitgang

Met CODE/INTERRUPT ACKNOWLEDGE wordt onderscheid gemaakt tussen instructie ophaal-cycli en geheugen-leescycli. Tevens worden interrupt-acknowledge cycli onderscheiden van I/O-cycli. COD/$\overline{\text{INTA}}$ zweeft gedurende bus hold-acknowledge.

$\overline{\text{LOCK}}$

uitgang (aktief-LAAG)

Het BUS LOCK-signaal geeft aan dat het andere systeem-busmasters verboden is om na de lopende buscyclus controle over de systeembus te krijgen. Het $\overline{\text{LOCK}}$-signaal kan expliciet worden geaktiveerd door de "LOCK" instructie-prefix of automatisch door 80L286 hardware tijdens geheugen XCHG-instructies, interrupt-acknowledge of toegang tot de sleutelwoordentabel. $\overline{\text{LOCK}}$ is aktief-LAAG en zwevend gedurende bus-hold acknowledge.

COD/$\overline{\text{INTA}}$	M/$\overline{\text{IO}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$	Bus cycle initiated
0 (Low)	0	0	0	Interrupt acknowledge
0	0	0	1	Reserved
0	0	1	0	Reserved
0	0	1	1	None; not a status cycle
0	1	0	0	IF A1 = 1 then halt; else shutdown
0	1	0	1	Memory data read
0	1	1	0	Memory data write
0	1	1	1	None; not a status cycle
1 (High)	0	0	0	Reserved
1	0	0	1	I/O Read
1	0	1	0	I/O Write
1	0	1	1	None; not a status cycle
1	1	0	0	Reserved
1	1	0	1	Memory instruction read
1	1	1	0	Reserved
1	1	1	1	None; not a status cycle

Tabel 7/3.5-2: Definities van de 80L286 buscyclus-status.

$\overline{\text{READY}}$

ingang (aktief-LAAG)

Met BUS READY wordt een buscyclus beëindigd. Buscycli worden onbegrensd verlengd totdat zij worden gestopt door het LAAG gaan van $\overline{\text{READY}}$. $\overline{\text{READY}}$ is een aktief-LAGE ingang die aan de systeemclock gerefereerde setup- en houdtijden nodig heeft om correct te werken. $\overline{\text{READY}}$ wordt gedurende bus hold-acknowledge genegeerd.

HOLD, HLDA

ingang/uitgang (aktief-HOOG)

BUS HOLD REQUEST en HOLD ACKNOWLEDGE regelen het bezit van de lokale 80L286-bus. Via de HOLD-ingang kan een

3.5 80L286

andere lokale busmaster om controle van de lokale bus vragen. Als dit is toegestaan maakt de 80L286 zijn busdrivers zwevend (3-state "uit") waarna HLDA wordt geaktiveerd, waardoor de bus hold-acknowledge conditie wordt bereikt. De lokale bus blijft aan de vragende master toegewezen totdat HOLD niet-aktief wordt. Hierdoor wordt HLDA gedeactiveerd en neemt de 80L286 de besturing van de lokale bus weer over waardoor de bus hold-acknowledge conditie wordt beëindigd. HOLD mag asynchroon zijn ten opzichte van de systeemclock. Beide signalen zijn aktief-HOOG.

INTR
ingang (aktief-HOOG)
Het INTERRUPT REQUEST-signaal vraagt de 80L286 om de uitvoering van het lopende programma uit te stellen en eerst service te verlenen aan een extern verzoek. Interrupt-requests worden gemaskeerd als het interrupt-enable bit in het flag-woord is gecleared. Wanneer de 80L286 op een interrupt-request reageert, worden twee interrupt-acknowledge buscycli uitgevoerd om een 8 bit interrupt-vector in te lezen die de herkomst van de interruptie aangeeft. Om te garanderen dat het programma echt wordt onderbroken moet INTR aktief blijven totdat de eerste interrupt-acknowledge cyclus is volbracht. INTR wordt aan het begin van elke processorcyclus afgetast en moet tenminste twee processorcycli vóór beëindiging van de lopende instructie aktief-HOOG zijn om vóór de volgende instructie te kunnen interrumperen.
INTR is niveau-gevoelig, aktief-HOOG en kan asynchroon met de systeemclock optreden.

NMI
ingang (aktief-HOOG)
Het NON-MASKABLE INTERRUPT REQUEST-signaal onderbreekt de 80L286 met een intern geleverde vectorwaarde van 2. Er worden geen interrupt-acknowledge cycli uitgevoerd. De interrupt-enable bit in het flag-woord van de 80L286 beïnvloedt deze ingang niet. Het NMI-signaal is aktief-HOOG, mag asynchroon zijn ten opzichte van de systeemclock en wordt na interne synchronisatie flankgetriggerd.
Om goed herkend te worden moet het signaal eerst tenminste vier systeem-clockcycli LAAG zijn geweest, waarna het zeker vier systeem-clockcycli HOOG moet blijven.

PEREQ, PEACK
ingang/uitgang (aktief-HOOG)
Met de PROCESSOR EXTENSION OPERAND REQUEST-ingang en de PROCESSOR EXTENSION ACKNOWLEDGE-uitgang worden de mogelijkheden van de 80L286 met betrekking tot geheugen-management en beveiliging naar processor-uitbreidingen vergroot.
Het PEREQ-signaal vraagt de 80L286 een data operand-transfer voor een processor-uitbreiding uit te voeren. De PEACK-uitgang laat de processor-uitbreiding weten wanneer de gevraagde operand wordt overgebracht.

PEREQ is aktief-HOOG en mag asynchroon zijn ten opzichte van de systeemclock, terwijl PEACK aktief-LAAG is.

BUSY, ERROR
ingangen (aktief-LAAG)
PROCESSOR EXTENSION BUSY- en ERROR geven de 80L286 informatie over de bedrijfsconditie van een processor-uitbreiding.
Een aktief BUSY-signaal laat de 80L286 op WAIT en sommige ESC-instructies stoppen met de uitvoering van een programma totdat BUSY weer niet-aktief (HOOG) wordt. De 80L286 kan, terwijl gewacht wordt op het niet-aktief worden van BUSY worden geïnterrumpeerd. Een aktief ERROR-signaal zorgt ervoor dat de 80L286 een processor-uitbreidingsinterrupt geeft bij het uitvoeren van WAIT of bepaalde ESC-instructies. Deze ingangen zijn aktief-LAAG en mogen asynchroon ten opzichte van de systeemclock werken.

3.5 80L286

RESET
ingang (aktief-HOOG)
SYSTEM RESET maakt de interne logika van de 80L286 leeg en is aktief-HOOG. De 80L286 kan op elk willekeurig moment opnieuw worden geïnitialiseerd door een LAAG-naar-HOOG overgang van het RESET-signaal als dat langer dan 16 systeemclockcycli aktief blijft.
Wanneer RESET aktief is komen de uitgangen van de 80L286 in de in tabel 7/3.5-3 aangegeven toestanden.

80L286 Pin State during Reset

Pin Value	Pin Names
1 (High)	$\overline{S0}$, $\overline{S1}$, $\overline{PEACK}$, A23–A0, $\overline{BHE}$, $\overline{LOCK}$
0 (Low)	M/$\overline{IO}$, COD/$\overline{INTA}$, HLDA
Three-state OFF	D15–D0

Tabel 7/3.5-3: Toestanden van de pennen tijdens het resetten van de 80L286.

De 80L286 begint te werken na een HOOG-naar-LAAG overgang op RESET. Deze overgang moet synchroon zijn met de systeemclock. De 80L286 heeft ongeveer 50 systeemclock-cycli nodig voor interne initialisatie voordat de eerste buscyclus wordt uitgevoerd (ophalen van code uit het power-on executie-adres).
Door een synchroon met de systeemclock optredende LAAG-naar-HOOG overgang van RESET wordt een nieuwe processorcyclus gestart op de volgende HOOG-naar-LAAG overgang van de systeemclock. De LAAG-naar-HOOG overgang van RESET mag asynchroon met de systeemclock optreden, maar dan kan niet voorspeld worden welke fase van de processorclock zal optreden gedurende de volgende systeem-periode. Synchrone LAAG-naar-HOOG overgangen zijn alleen nodig in systemen waarbij de processorclock fase-synchroon moet zijn met een andere clock.

V_{SS} (GND)
ingang
De GROUND-ingang wordt verbonden met de systeem-massa (aarde: 0 V).

V_{CC}
ingang
De SYSTEM POWER-ingang wordt aangesloten op de +5 V systeemvoeding (+/-5 %).

CAP
ingang (aktief-HOOG)
Een SUBSTRATE FILTER CONDENSATOR van 0,047 F (+/-20 %, 12 V) moet tussen deze pen en aarde worden aangesloten. Deze condensator filtert het uitgangssignaal van de inwendige substraat bias-generator. De condensator mag een maximale DC lekstroom van 1 μA hebben.
Om correct te kunnen werken moet de substraat bias-generator van de 80L286 deze condensator tot de werkspanning opladen. De oplaadtijd van de condensator bedraagt maximaal 5 ms nadat V_{CC} en CLK hun gespecificeerde DC- en AC-parameters hebben bereikt. Gedurende deze tijd mag een RESET worden gegeven om verkeerd werken van de CPU te voorkomen. Na deze tijd kan de fase van de processorclock van de 80L286 worden gesynchroniseerd op een andere clock door RESET synchroon met de systeemclock LAAG te pulseren.

Overige kenmerken

Voor de werking van registers, status en instructieset wordt verwezen naar de 80286 microprocessor. In de tabellen 7/3.5-4 tot en met -8 en de figuren 7/3.5-3 tot en met -8 zijn de essentiële elektrische en timing-karakteristieken van de 80L286 vermeld.

3.5 80L286

ABSOLUTE MAXIMUM RATINGS

Storage Temperature .... −65°C to +150°C
Voltage on Any Pin with
Respect to Ground .... −1.0 V to 7.0 V
Power Dissipation .... 2.89 Watts

Tabel 7/3.5-4: Maximaal toegelaten waarden.

OPERATING RANGES

Operating Voltage Range .... +4.75 V to +5.25 V
Case Operating Temperature Range . . 0°C to +85°C

Tabel 7/3.5-5: Aanbevolen bedrijfscondities van de 80L286.

(T_{CASE} = 0°C to 85°C, V_{CC} = 5 V±5%)

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
V_{IL}	Input Low Voltage		−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC} + 0.5	V
V_{ILC}	CLK Input Low Voltage		−0.5	.6	V
V_{IHC}	CLK Input High Voltage		3.8	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage	I_{OL} = 2.0 mA		0.45	V
V_{OH}	Output High Voltage	I_{OH} = −400 µA	2.4		V
I_{LI}	Input Leakage Current	0 V ≤ V_{IN} ≤ V_{CC}		± 10	µA
I_{LO}	Output Leakage Current	0.45 V ≤ V_{OUT} ≤ V_{CC}		± 10	µA
I_{CC}	Supply Current	T_C = 0°C T_C = 85°C		550 475	mA mA
C_{CLK}	CLK Input Capacitance	F_C = 1 MHz		20	pF
C_{IN}	Other Input Capacitance	F_C = 1 MHz		10	pF
C_O	Input/Output Capacitance	F_C = 1 MHz		20	pF
I_{LO}	Output Leakage Current	0 V ≤ V_{OUT} < 0.45 V		± 1	mA
I_{IL}	Input Sustaining Current on $\overline{BUSY}$ and $\overline{ERROR}$ pins	V_{IN} = 0 V	30	500	µA
I_{LCR}	Input CLK Leakage Current	0.45 ≤ V_{IN} ≤ V_{CC}		± 10	µA
I_{LCR}	Input CLK Leakage Current	0 V ≤ V_{IN} ≤ 0.45 V		± 1	mA

Tabel 7/3.5-6: Gelijkspanningskarakteristieken van de 80L286.

3.5 80L286

V_{CC} = +5 V ±5%, T_{CASE} = 0°C to +85°C
AC timings are referenced to 0.8 V and 2.0 V points of the signals as illustrated in data sheet waveforms, unless otherwise noted.

Parameter Symbol	Parameter Description	Test Conditions	8 MHz		10 MHz		Unit
			Min.	Max.	Min.	Max.	
1	System Clock (CLK) Period		62	125	50	125	ns
2	System Clock (CLK) Low Time	@ 1.0 V	15	100	12	109	ns
3	System Clock (CLK) High Time	@ 3.6 V	25	110	16	113	ns
17	System Clock (CLK) RISE Time	1.0 V to 3.6 V		10		8	ns
18	System Clock (CLK) FALL Time	3.6 V to 1.0 V		10		8	ns
4	Asynchronous Inputs SETUP Time	(Note 1)	20		20		ns
5	Asynchronous Inputs HOLD Time	(Note 1)	20		20		ns
6	RESET SETUP Time		28		23		ns
7	RESET HOLD Time		5		5		ns
8	Read Data SETUP Time		10		8		ns
9	Read Data HOLD Time		8		8		ns
10	$\overline{\text{READY}}$ SETUP Time		38		26		ns
11	$\overline{\text{READY}}$ HOLD Time		25		25		ns
12	Status/$\overline{\text{PEACK}}$ Valid Delay	(Notes 2, 3)	1	40	–	–	ns
12A	Status/$\overline{\text{PEACK}}$ Active Delay	(Notes 2, 3)	–	–	1	22	ns
12B	Status/$\overline{\text{PEACK}}$ Inactive Delay	(Notes 2, 3)	–	–	1	30	ns
13	Address Valid Delay	(Notes 2, 3)	1	60	1	35	ns
14	Write Data Valid Delay	(Notes 2, 3)	0	50	0	30	ns
15	Address/Status/Data Float Delay	(Notes 2, 4)	0	50	0	47	ns
16	HLDA Valid Delay	(Notes 2, 3)	0	50	0	47	ns
19	Address Valid to Status SETUP Time	(Notes 3, 5, 6)	38		27		ns

Tabel 7/3.5-7: Schakeltijden van de 80L286 bij 8 MHz en 10 MHz.

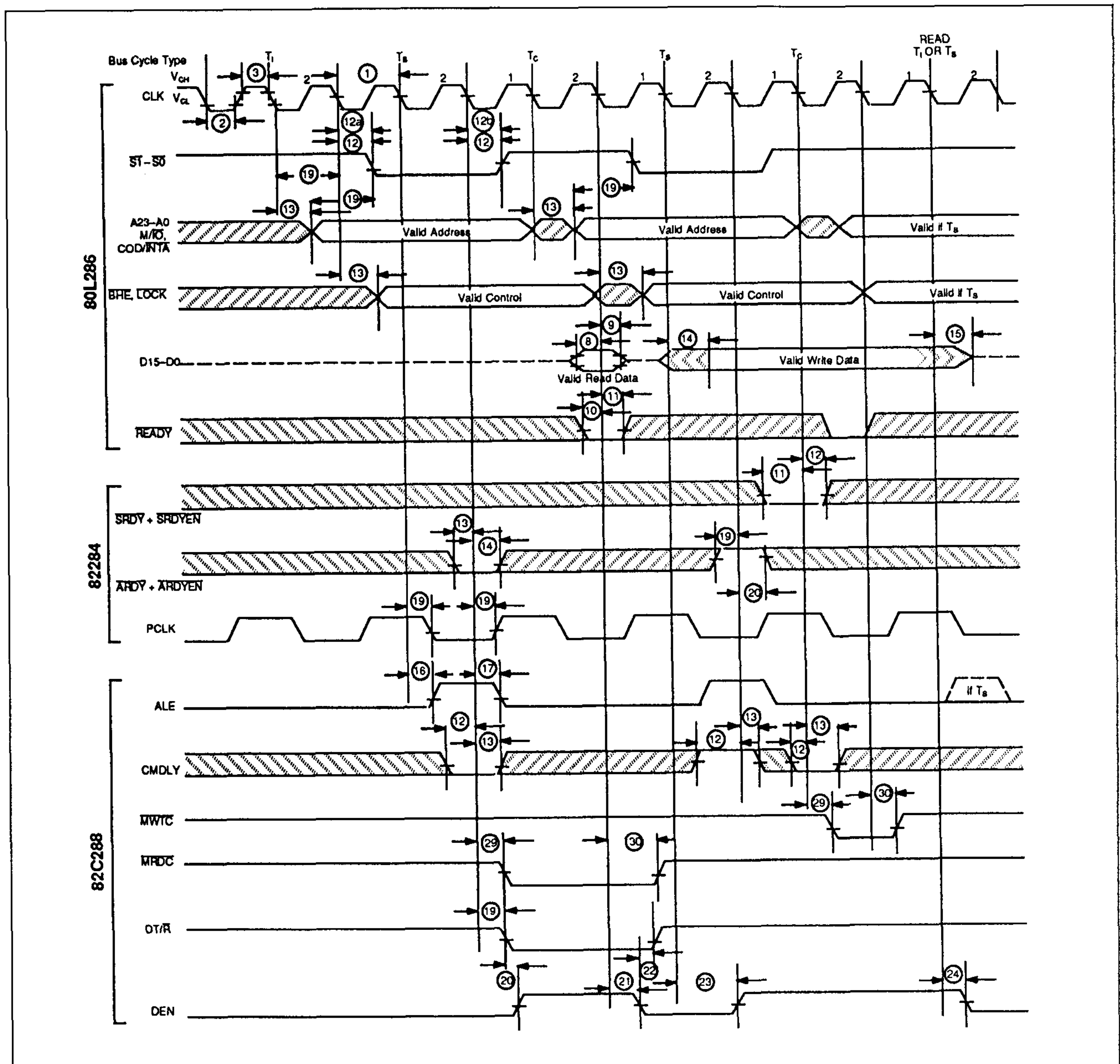
3.5 80L286

Parameter Symbol	Parameter Description	Test Conditions	12.5 MHz Min.	12.5 MHz Max.	16 MHz Min.	16 MHz Max.	Unit
1	System Clock (CLK) Period		40	125	31	125	ns
2	System Clock (CLK) Low Time	@ 1.0 V	11	112	10	113	ns
3	System Clock (CLK) High Time	@ 3.6 V	13	114	12	115	ns
17	System Clock (CLK) RISE Time	1.0 V to 3.6 V		8		5	ns
18	System Clock (CLK) FALL Time	3.6 V to 1.0 V		8		4	ns
4	Asynchronous Inputs SETUP Time	(Note 1)	15		11		ns
5	Asynchronous Inputs HOLD Time	(Note 1)	15		11		ns
6	RESET SETUP Time		18		14		ns
7	RESET HOLD Time		5		3		ns
8	Read Data SETUP Time		5		5		ns
9	Read Data HOLD Time		6		5		ns
10	$\overline{\text{READY}}$ SETUP Time		22		15		ns
11	$\overline{\text{READY}}$ HOLD Time		20		15		ns
12	Status/$\overline{\text{PEACK}}$ Valid Delay	(Notes 2, 3)	–	–	–	–	ns
12A	Status/$\overline{\text{PEACK}}$ Active Delay	(Notes 2,3)	3	18	1	18	ns
12B	Status/$\overline{\text{PEACK}}$ Inactive Delay	(Notes 2,3)	3	20	1	20	ns
13	Address Valid Delay	(Notes 2, 3)	1	32	1	29	ns
14	Write Data Valid Delay	(Notes 2, 3)	0	30	0	22	ns
15	Address/Status/Data Float Delay	(Notes 2, 4)	0	32	0	29	ns
16	HLDA Valid Delay	(Notes 2, 3)	0	25	0	25	ns
19	Address Valid to Status SETUP Time	(Notes 3, 5, 6)	22		22		ns

Notes:
1. Asynchronous inputs are INTR, NMI, HOLD, PEREQ, $\overline{\text{ERROR}}$, and $\overline{\text{BUSY}}$. This specification is given only for testing purposes, to assure recognition at a specific CLK edge.
2. Delay from 1.0 V on the CLK to 0.8 V or 2.0 V or float on the output as appropriate for valid or floating condition.
3. Output load: $C_L = 100$ pF.
4. Float condition occurs when output current is less than I_{LO} in magnitude.
5. Delay measured from address either reaching 0.8 V or 2.0 V (valid) to status going active reaching 2.0 V or status going inactive reaching 0.8 V.
6. For load capacitance of 10 pF on STATUS/PEACK lines, subtract typically 7 ns for 8 MHz spec, and maximum 7 ns for 10 MHz spec.

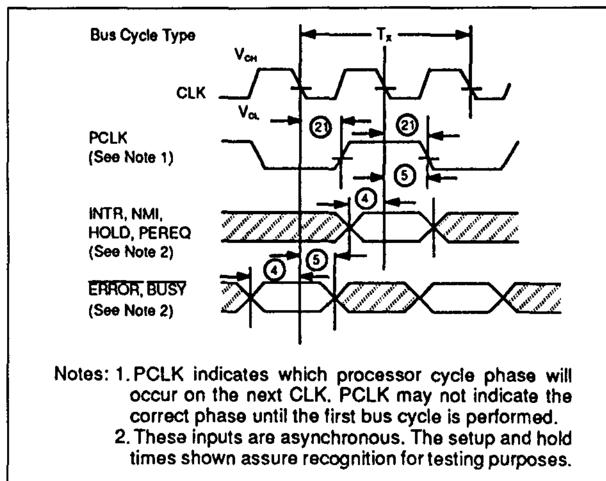
Tabel 7/3.5-8: Schakeltijden van de 80L286 bij 12,5 en 16 MHz.

3.5 80L286

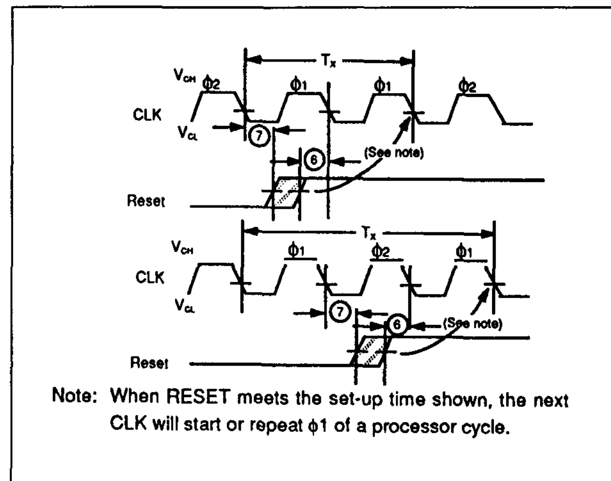


Figuur 7/3.5-3: De belangrijkste schakeltijden van de 80L286, de clockdriver 82284 en de buscontroller 82C288 (zie ook de tabellen 7/3.5-7 en -8).

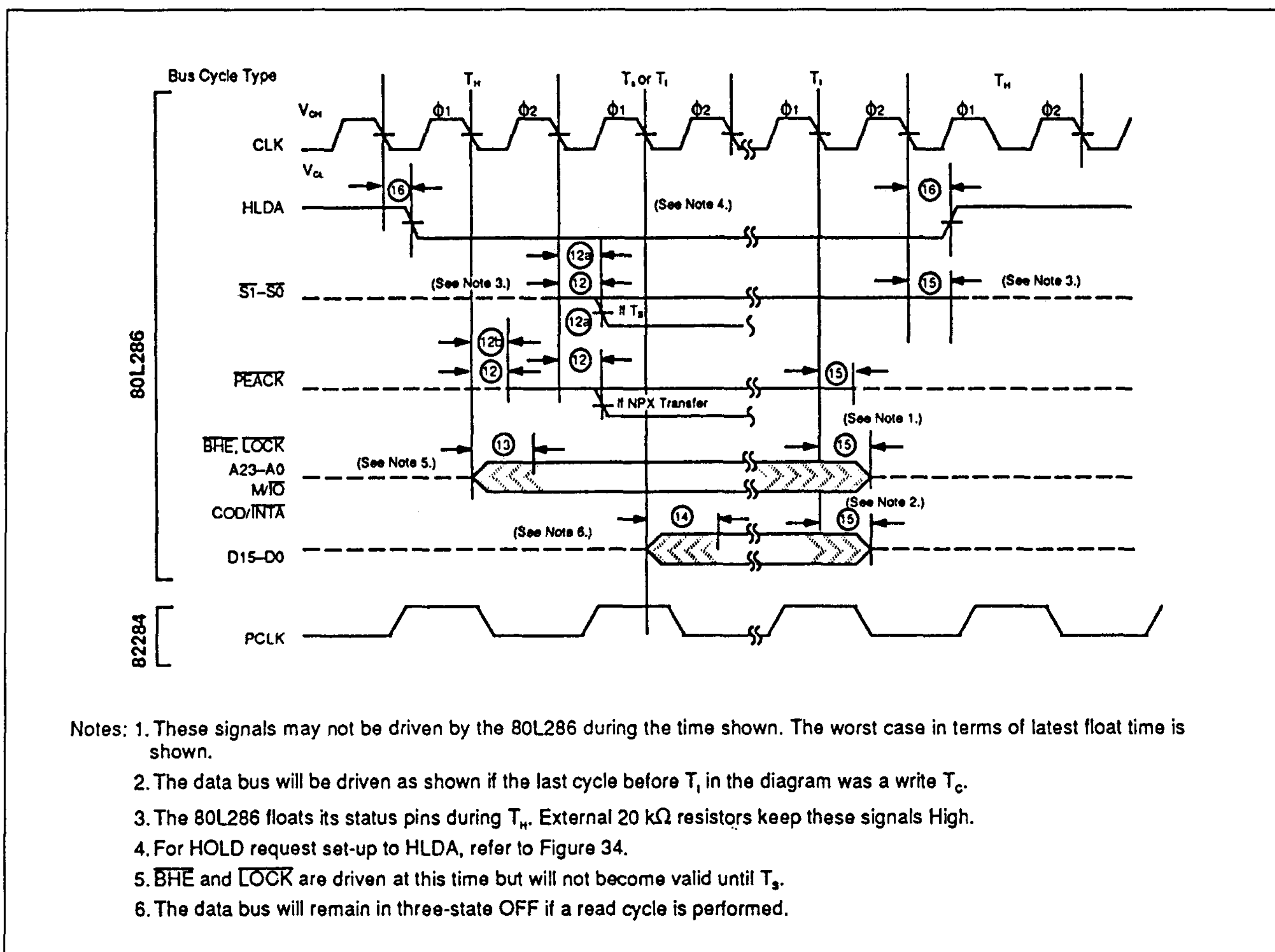
3.5 80L286



Figuur 7/3.5-4: Timing van asynchrone ingangssignalen.

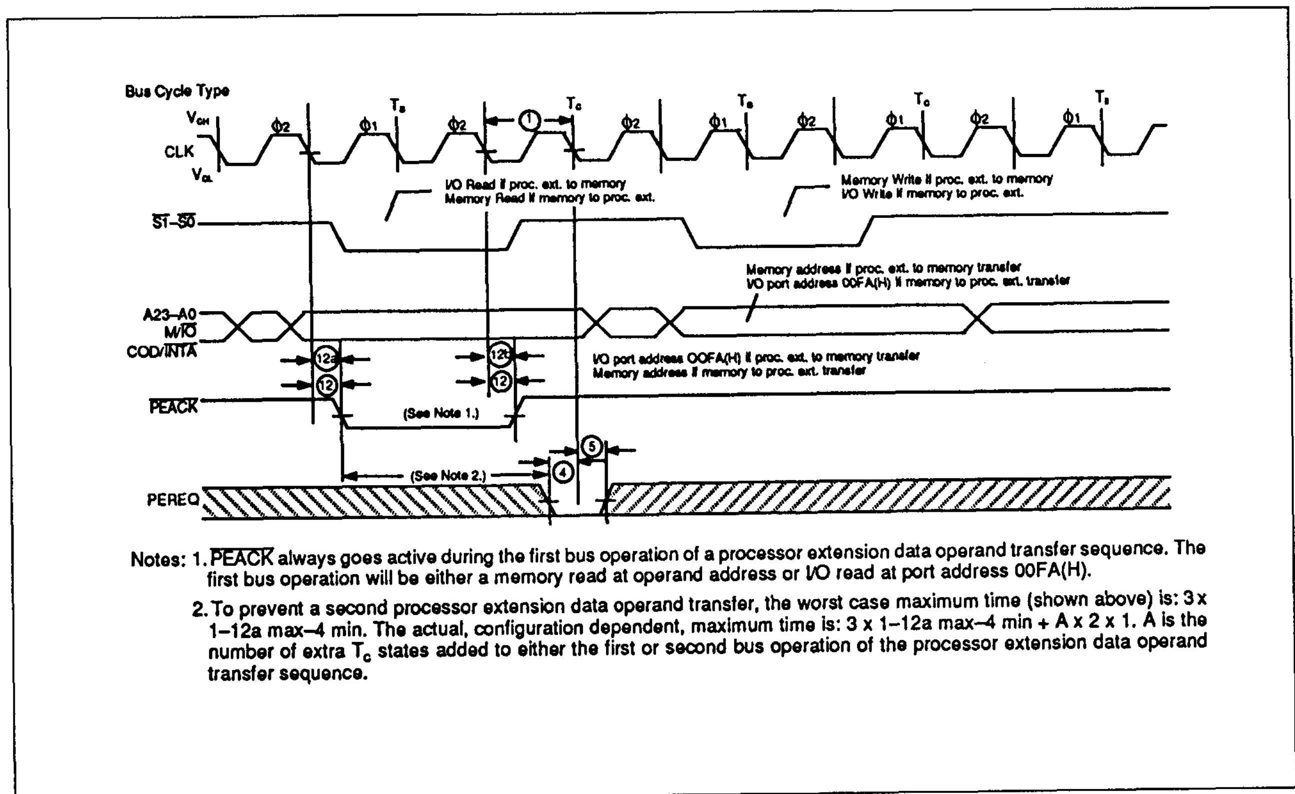


Figuur 7/3.5-5: Timing van reset en fase van de volgende processorcyclus.



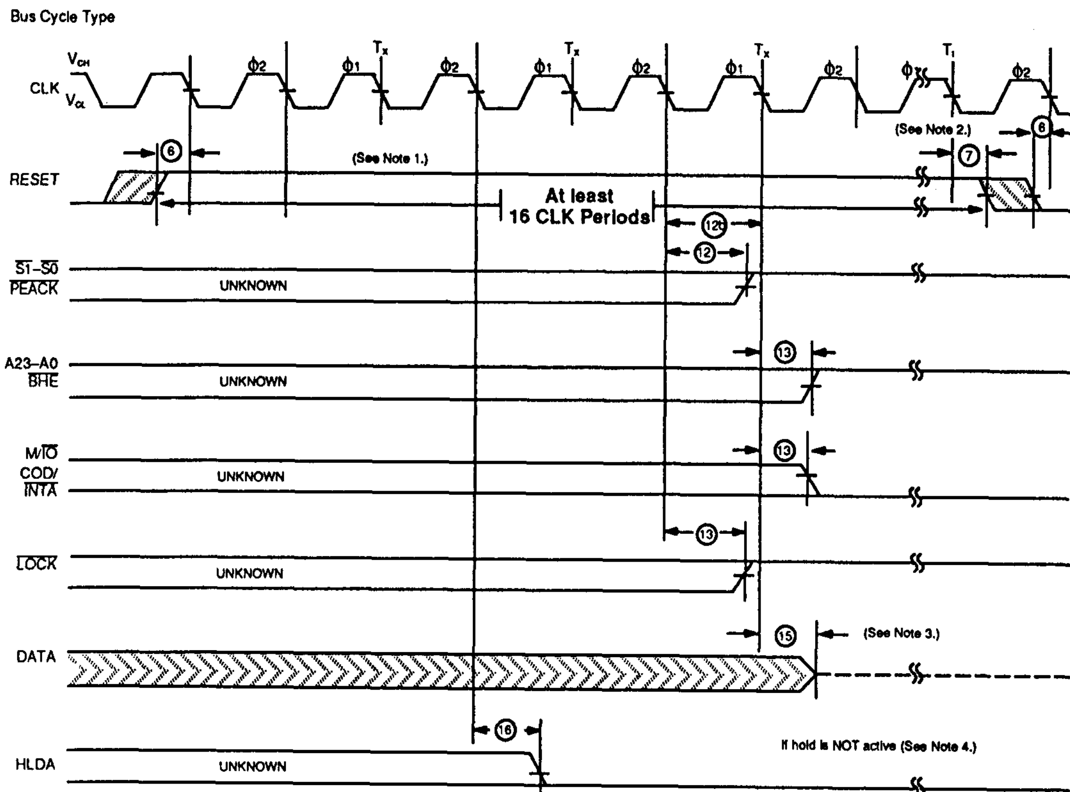
Figuur 7/3.5-6: Het oproepen van de hold-situatie.

3.5 80L286



Figuur 7/3.5-7: Timing van PEREQ/PEACK.

3.5 80L286



Notes: 1. Set-up time for RESET ↑ may be violated with the consideration that ϕ 1 of the processor clock may begin one system CLK period later.

2. Set-up and hold times for RESET ↓ must be met for proper operation, but RESET ↓ may occur during ϕ 1 or ϕ 2.

3. The data bus is only guaranteed to be in three-state OFF at the time shown.

4. HOLD is acknowledged during RESET, causing HLDA to go active and the appropriate pins to float. If HOLD remains active while RESET goes inactive, the 80L286 remains in HOLD state and will not perform any bus accesses until HOLD is deactivated.

Figuur 7/3.5-8: Initiële toestand van de signalen van de 80L286 tijdens resetten.

7/4

Tweeëndertig bits processoren

Inhoud

7/4.1

80386

Inleiding

Algemeen

De 80386 is een "high performance" 32 bit microprocessor voor geavanceerde toepassingen.

Deze opvolger van de 80286 wordt bijvoorbeeld gebruikt voor grafische toepassingen met hoge resolutie, desk top publishing (DTP), spraak- en patroonherkenning, in CAE/CAD werkstations en bij de automatisering van kantoren en fabrieken. De 80386 is oorspronkelijk van Intel afkomstig, maar wordt tegenwoordig ook door andere fabrikanten, zoals AMD vervaardigd. De 80386 bevat 275.000 transistoren op één chip en is alleen in CMOS verkrijgbaar: voor de eerste exemplaren die op 12 en 16 MHz werkten werd CHMOS III toegepast, voor de nieuwere het CHMOS IV-proces dat nog kleinere afmetingen en hogere snelheden toelaat (20, 25 en 33 MHz).

De 80386 is verkrijgbaar in twee uitvoeringen (die beide inwendig een 32 bit architectuur hebben): de 80386DX en de 80386SX. De DX is de compleetste met 32 bit registers, adressen en data.

De SX werkt extern met een 16 bit databus en een 24 bit adresbus. De SX-versie zal dus vaak goedkopere oplossingen bieden als 16 bit hardwaresystemen voldoende zijn. Van beide typen zijn nu ook Low Voltage uitvoeringen verkrijgbaar. Hierna wordt eerst de 80386DX behandeld, waarna de overige versies aan bod komen. De uitzonderingen van een bepaalde versie worden daarbij nader toegelicht.

Opmerking

Overigens heeft Intel ook al de 386SL SuperSet uitgebracht: een uit twee VLSI-componenten bestaande set, speciaal voor draagbare computers zoals "palmtop" en "notebook" tot en met volledig ingerichte "laptop's".

Daarnaast wordt de 386 in een speciale EX en CX uitvoering ook toegepast in "embedded" systemen. Deze worden bij de microcontrollers behandeld.

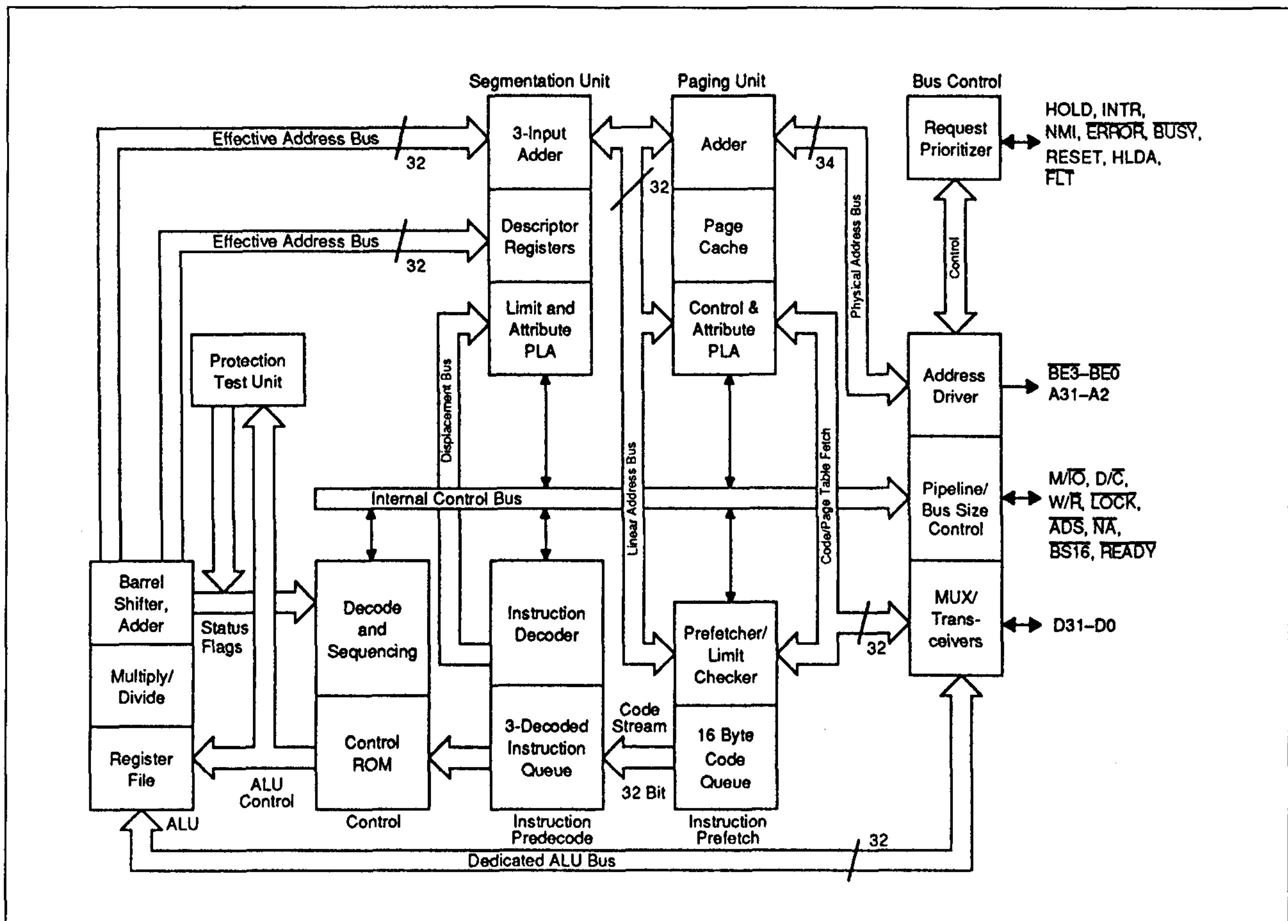
Kenmerken

De 80386 biedt vele nieuwe, krachtige mogelijkheden zoals 3 à 4 miljoen instructies per seconde, een complete 32 bit architectuur, een fysieke adresruimte van 4 Gigabytes en 64 Terabytes virtueel geheugen. Ondanks deze vernieuwingen kan de enorme hoeveelheid software die voor de voorgangers 8086 en 80286 is geschreven ook voor de 80386 worden gebruikt: de 80386 is "object-code compatibel" met alle leden van de 8086-familie. De 80386 heeft "virtuele machine" mogelijkheden die van groot belang zijn, omdat de processor hierdoor gemakkelijk kan omschakelen tussen programma's die onder verschillende bedrijfssystemen werken, zoals Unix en MS-DOS.

Algemene gegevens

- flexibele 32-bit microprocessor
- data-typen: 8, 16 en 32 bit
- 8 algemene 32 bit registers
- grote adresruimte: 4 Gbyte fysiek, 64 Tbyte virtueel, maximum 4 Gbyte segment-afmeting

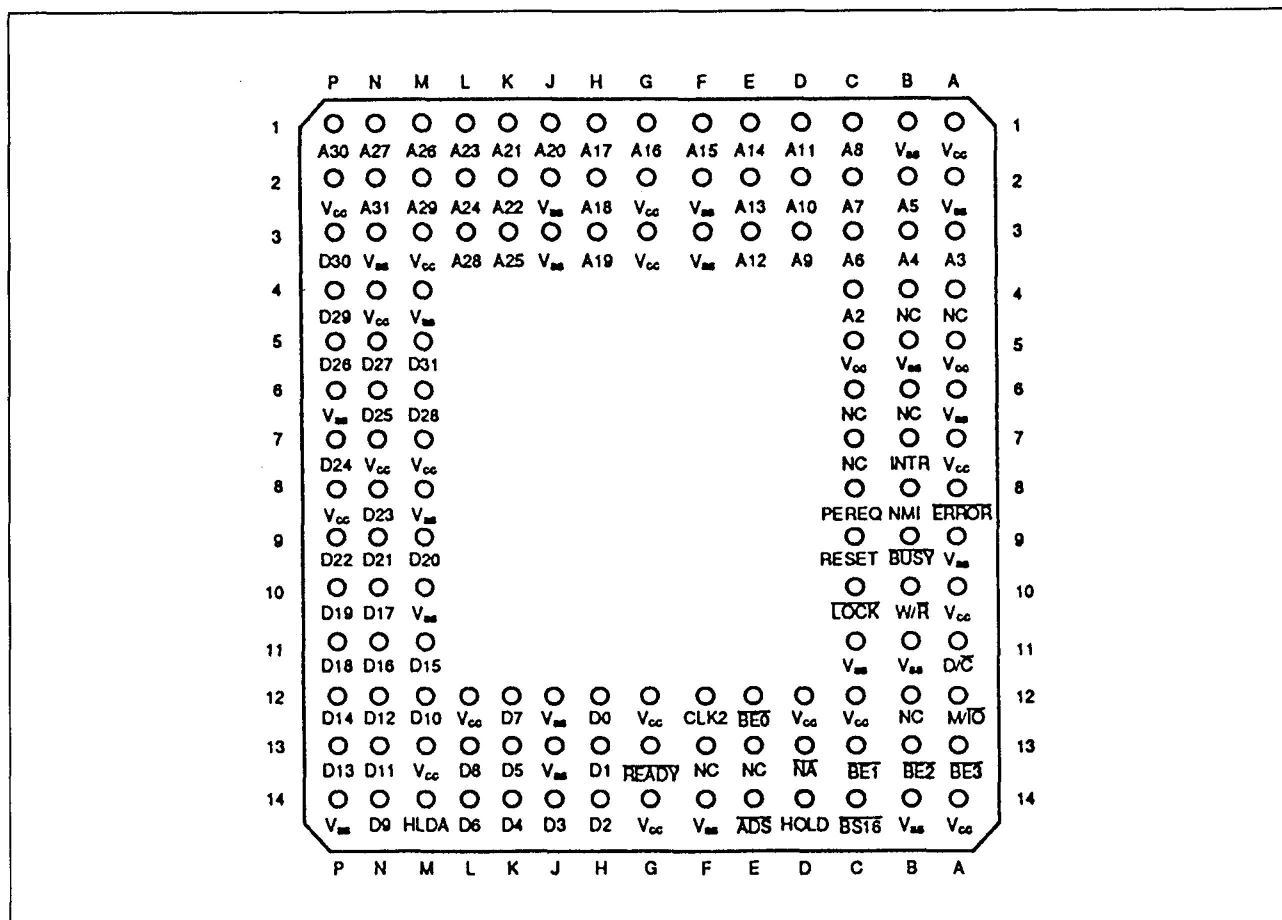
4.1 80386



Figuur 7/4.1-1: Blokschema van de 80386DX microprocessor.

- geïntegreerde memory management unit (MMU), ondersteuning van virtueel geheugen, geheugenbescherming op vier niveaus
- object-code compatibel met alle xx86 microprocessoren
- virtuele 8086 mode voor het werken met 8086 software
- ondersteuning van hardware debugging
- gepijplijnde uitvoering van instructies
- kloksnelheden: (12, 16), 20, 25, 33 MHz
- bus-bandbreedte: 40, 50, 66 Mbytes/sec
- numerieke ondersteuning door 80387DX coprocessor
- behuizing: 132-pens Pin Grid Array (PGA, figuur 7/4.1-2), AMD-type Am386DX ook in 132-pens Plastic Quad Flat Pack PQFP, figuur 7/4.1-3
- fabrikanten: Intel, AMD

4.1 80386



Figuur 7/4.1-2: Bovenaanzicht van de 132-pens PGA-uitvoering van de 80386DX microprocessor.

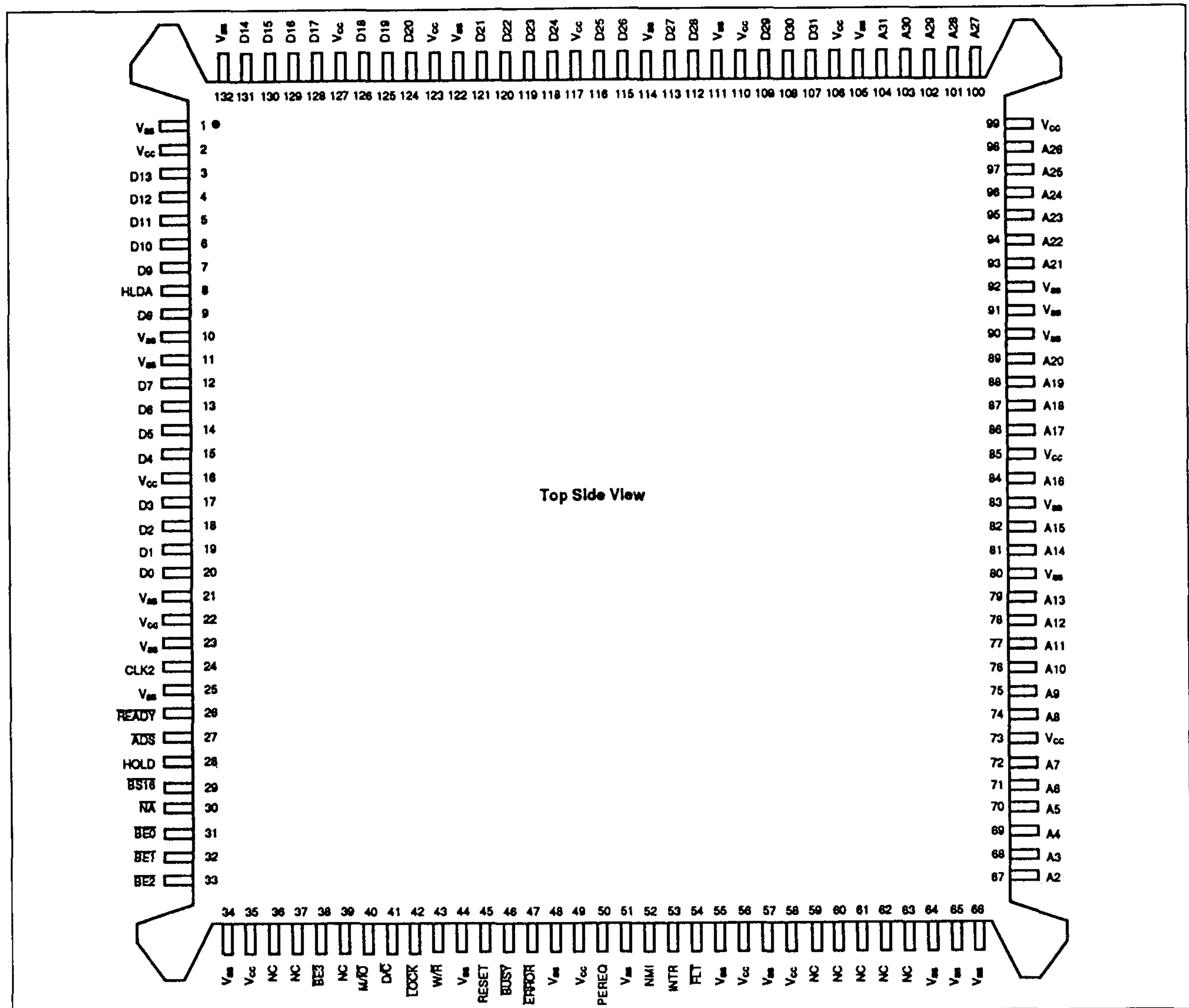
4.1 80386

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A2	C4	A24	L2	D6	L14	D28	M6	V_{cc}	C12	V_{ss}	F2
A3	A3	A25	K3	D7	K12	D29	P4		D12		F3
A4	B3	A26	M1	D8	L13	D30	P3		G2		F14
A5	B2	A27	N1	D9	N14	D31	M5		G3		J2
A6	C3	A28	L3	D10	M12	D/$\overline{C}$	A11		G12		J3
A7	C2	A29	M2	D11	N13	$\overline{ERROR}$	A8		G14		J12
A8	C1	A30	P1	D12	N12	HLDA	M14		L12		J13
A9	D3	A31	N2	D13	P13	HOLD	D14		M3		M4
A10	D2	$\overline{ADS}$	E14	D14	P12	INTR	B7		M7		M8
A11	D1	$\overline{BE0}$	E12	D15	M11	$\overline{LOCK}$	C10		M13		M10
A12	E3	$\overline{BE1}$	C13	D16	N11	M/$\overline{IO}$	A12		N4		N3
A13	E2	$\overline{BE2}$	B13	D17	N10	$\overline{NA}$	D13		N7		P6
A14	E1	$\overline{BE3}$	A13	D18	P11	NMI	B8		P2		P14
A15	F1	$\overline{BS16}$	C14	D19	P10	PEREQ	C8		P8	W/$\overline{R}$	B10
A16	G1	$\overline{BUSY}$	B9	D20	M9	$\overline{READY}$	G13	V_{ss}	A2	NC	A4
A17	H1	CLK2	F12	D21	N9	RESET	C9		A6		B4
A18	H2	D0	H12	D22	P9	V_{cc}	A1		A9		B6
A19	H3	D1	H13	D23	N8		A5		B1		B12
A20	J1	D2	H14	D24	P7		A7		B5		C6
A21	K1	D3	J14	D25	N6		A10		B11		C7
A22	K2	D4	K14	D26	P5		A14		B14		E13
A23	L1	D5	K13	D27	N5		C5		C11		F13

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
A1	V_{cc}	B9	$\overline{BUSY}$	D3	A9	H1	A17	L13	D8	N7	V_{cc}
A2	V_{ss}	B10	W/$\overline{R}$	D12	V_{cc}	H2	A18	L14	D6	N8	D23
A3	A3	B11	V_{ss}	D13	$\overline{NA}$	H3	A19	M1	A26	N9	D21
A4	NC	B12	NC	D14	HOLD	H12	D0	M2	A29	N10	D17
A5	V_{cc}	B13	$\overline{BE2}$	E1	A14	H13	D1	M3	V_{cc}	N11	D16
A6	V_{ss}	B14	V_{ss}	E2	A13	H14	D2	M4	V_{ss}	N12	D12
A7	V_{cc}	C1	A8	E3	A12	J1	A20	M5	D31	N13	D11
A8	$\overline{ERROR}$	C2	A7	E12	$\overline{BE0}$	J2	V_{ss}	M6	D28	N14	D9
A9	V_{ss}	C3	A6	E13	NC	J3	V_{ss}	M7	V_{cc}	P1	A30
A10	V_{cc}	C4	A2	E14	$\overline{ADS}$	J12	V_{ss}	M8	V_{ss}	P2	V_{cc}
A11	D/$\overline{C}$	C5	V_{cc}	F1	A15	J13	V_{ss}	M9	D20	P3	D30
A12	M/$\overline{IO}$	C6	NC	F2	V_{ss}	J14	D3	M10	V_{ss}	P4	D29
A13	$\overline{BE3}$	C7	NC	F3	V_{ss}	K1	A21	M11	D15	P5	D26
A14	V_{cc}	C8	PEREQ	F12	CLK2	K2	A22	M12	D10	P6	V_{ss}
B1	V_{ss}	C9	RESET	F13	NC	K3	A25	M13	V_{cc}	P7	D24
B2	A5	C10	$\overline{LOCK}$	F14	V_{ss}	K12	D7	M14	HLDA	P8	V_{cc}
B3	A4	C11	V_{ss}	G1	A16	K13	D5	N1	A27	P9	D22
B4	NC	C12	V_{cc}	G2	V_{cc}	K14	D4	N2	A31	P10	D19
B5	V_{ss}	C13	$\overline{BE1}$	G3	V_{cc}	L1	A23	N3	V_{ss}	P11	D18
B6	NC	C14	$\overline{BS16}$	G12	V_{cc}	L2	A24	N4	V_{cc}	P12	D14
B7	INTR	D1	A11	G13	$\overline{READY}$	L3	A28	N5	D27	P13	D13
B8	NMI	D2	A10	G14	V_{cc}	L12	V_{cc}	N6	D25	P14	V_{ss}

Tabel 7/4.1-1: Pen-functies van de PGA-versie van de 80386DX. Boven functioneel gegroepeerd, onder gesorteerd volgens pen-nummer.

4.1 80386



Figuur 7/4.1-3: Bovenaanzicht van de 132-pens PQFP-uitvoering van de Am80386DX microprocessor.

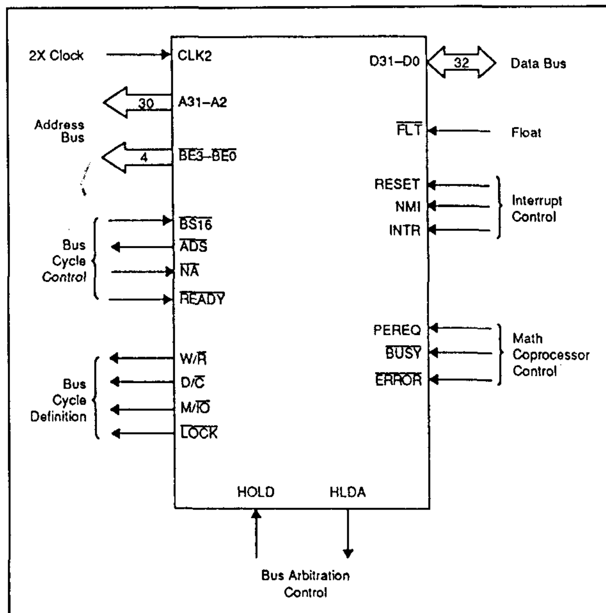
4.1 80386

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
A2	67	A24	96	D6	13	D28	112	V_{cc}	56	V_{ss}	64
A3	68	A25	97	D7	12	D29	109		58		65
A4	69	A26	98	D8	9	D30	108		73		66
A5	70	A27	100	D9	7	D31	107		85		80
A6	71	A28	101	D10	6	D/$\overline{C}$	41		99		83
A7	72	A29	102	D11	5	$\overline{ERROR}$	47		106		90
A8	74	A30	103	D12	4	HLDA	8		110		91
A9	75	A31	104	D13	3	HOLD	28		117		92
A10	76	$\overline{ADS}$	27	D14	131	INTR	53		123		105
A11	77	$\overline{BE0}$	31	D15	130	$\overline{LOCK}$	42		127		111
A12	78	$\overline{BE1}$	32	D16	129	M/$\overline{IO}$	40	V_{ss}	1		114
A13	79	$\overline{BE2}$	33	D17	128	$\overline{NA}$	30		10		122
A14	81	$\overline{BE3}$	38	D18	126	NMI	52		11		132
A15	82	$\overline{BS16}$	29	D19	125	PEREQ	50		21	W/$\overline{R}$	43
A16	84	$\overline{BUSY}$	46	D20	124	$\overline{READY}$	26		23	NC	36
A17	86	CLK2	24	D21	121	RESET	45		25		37
A18	87	D0	20	D22	120	V_{cc}	2		35		39
A19	88	D1	19	D23	119		16		44		59
A20	89	D2	18	D24	118		22		48		60
A21	93	D3	17	D25	116		34		51		61
A22	94	D4	15	D26	115		49		55		62
A23	95	D5	14	D27	113	$\overline{FLT}$	54		57		63

Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name
1	V_{ss}	23	V_{ss}	45	RESET	67	A2	89	A20	111	V_{ss}
2	V_{cc}	24	CLK2	46	$\overline{BUSY}$	68	A3	90	V_{ss}	112	D28
3	D13	25	V_{ss}	47	$\overline{ERROR}$	69	A4	91	V_{ss}	113	D27
4	D12	26	$\overline{READY}$	48	V_{ss}	70	A5	92	V_{ss}	114	VSS
5	D11	27	$\overline{ADS}$	49	V_{cc}	71	A6	93	A21	115	D26
6	D10	28	HOLD	50	PEREQ	72	A7	94	A22	116	D25
7	D9	29	$\overline{BS16}$	51	V_{ss}	73	V_{cc}	95	A23	117	V_{cc}
8	HLDA	30	$\overline{NA}$	52	NMI	74	A8	96	A24	118	D24
9	D8	31	$\overline{BE0}$	53	INTR	75	A9	97	A25	119	D23
10	V_{ss}	32	$\overline{BE1}$	54	$\overline{FLT}$	76	A10	98	A26	120	D22
11	V_{ss}	33	$\overline{BE2}$	55	V_{ss}	77	A11	99	V_{cc}	121	D21
12	D7	34	V_{cc}	56	V_{cc}	78	A12	100	A27	122	V_{ss}
13	D6	35	V_{ss}	57	V_{ss}	79	A13	101	A28	123	V_{cc}
14	A5	36	NC	58	V_{cc}	80	V_{ss}	102	A29	124	D20
15	D4	37	NC	59	NC	81	A14	103	A30	125	D19
16	V_{cc}	38	$\overline{BE3}$	60	NC	82	A15	104	A31	126	D18
17	D3	39	NC	61	NC	83	V_{ss}	105	V_{ss}	127	V_{cc}
18	D2	40	M/$\overline{IO}$	62	NC	84	A16	106	V_{cc}	128	D17
19	D1	41	D/$\overline{C}$	63	NC	85	V_{cc}	107	D31	129	D16
20	D0	42	$\overline{LOCK}$	64	V_{ss}	86	A17	108	D30	130	D15
21	V_{ss}	43	W/$\overline{R}$	65	V_{ss}	87	A18	109	D29	131	D14
22	V_{cc}	44	V_{ss}	66	V_{ss}	88	A19	110	V_{cc}	132	V_{ss}

Tabel 7/4.1-2: Pen-functies van de PQFP-versie van de Am80386DX. Boven indeling volgens functie, onder sortering volgens pen-nummer.

4.1 80386



Figuur 7/4.1-4: Logisch symbool van de 80386.

Beschrijving van de penfuncties

Verwezen wordt naar het logisch symbool, figuur 7/4.1-4.

- CLK2, Clock (ingang)
  Levert de fundamentele timing voor de 80386 processor.
- D31 tot en met D0, Data Bus (ingangen/uitgangen)
  Ingangen voor data tijdens leescycli voor geheugen, I/O- en interrupt-acknowledge en uitgangen tijdens schrijfcycli voor geheugen- en I/O.
- A31 tot en met A2, Address Bus (uitgangen)
  Uitgangssignalen voor het bepalen van de adressen van fysiek geheugen en I/O-poorten.
- BE3 tot en met BE0, Byte Enables (aktief-LAGE uitgangen)
  Deze signalen geven aan welke databytes van de databus deelnemen aan een buscyclus.
- W/R, Write/Read (uitgang)
  Met dit signaal wordt onderscheid gemaakt tussen lees- en schrijfcycli.
- D/C, Data/Control (uitgang)
  Een pen die onderscheid maakt tussen datacycli van geheugen of I/O en besturingscycli, zoals interrupt acknowledge, halt of instruction fetching.
- M/IO, Memory I/O (uitgang)
  Met dit buscyclus definitie-signaal wordt onderscheid gemaakt tussen geheugencycli en input/output-cycli.
- LOCK, Bus Lock (aktief-LAGE uitgang)
  Een buscyclus definitie-signaal dat aangeeft dat het andere systeem-busmasters verboden is controle over de systeembus te verkrijgen terwijl die aktief is.
- ADS, Address Status (aktief-LAGE uitgang)
  Dit signaal geeft aan dat een geldige buscyclus-definitie en adres (W/R, D/C, M/IO, BE0, BE1, BE2, BE3 en A31 tot en met A2) op de pennen van de 80386 staan.
- NA, Next Address (aktief-LAGE ingang)
  Wordt gebruikt om te vragen een adres in de pijplijn te zetten.
- READY, Bus Ready (aktief-LAGE ingang)
  Met dit signaal wordt de buscyclus beëindigd.
- BS16, Bus Size 16 (aktief-LAGE ingang)
  Maakt directe aansluiting op 32 bit en 16 bit data-bussen mogelijk.
- HOLD, Bus Hold Request (aktief-HOGE ingang)
  Stelt een andere busmaster in staat om besturing van de lokale bus aan te vragen.
- HLDA, Bus Hold Acknowledge (aktief-HOGE uitgang)
  Dit signaal geeft aan dat de 80386 de besturing van zijn lokale bus heeft overgedragen aan een andere busmaster.
- BUSY, Busy (aktief-LAGE ingang)
  Signaleert een "bezig"-conditie van een processor-uitbreiding.
- ERROR, Error (aktief-LAGE ingang)
  Signaleert een fout-conditie van een processor-uitbreiding.
- FLT*, Float (aktief-LAGE ingang)
  Een ingangssignaal dat alle bidirectionele- en uitgangssignalen, inclusief HLDA, in de hoog-impedante 3-state toestand zet. Deze ingang is alleen aanwezig op het

4.1 80386

AMD-type Am386 vanaf versie CO of later (alleen PQFP).

- PEREQ, Processor Extension Request (aktief-HOGE ingang)
  Ingang die aangeeft dat de processoruitbreiding data heeft die door de 80386 moet worden overgebracht.
- INTR, Interrupt Request (aktief-HOGE ingang)
  Is een maskeerbare ingang waarmee de 80386 wordt gevraagd de uitvoering van het lopende programma uit te stellen om eerst een interrupt-acknowledge functie uit te voeren.
- NMI, Non-Maskable Interrupt Request (aktief-HOGE ingang)
  Een niet-maskeerbare ingang die de 80386 signaleert om de uitvoering van het lopende programma uit te stellen en eerst een interrupt-acknowledge functie uit te voeren.
- RESET, Reset (aktief-HOGE ingang)
  Onderbreekt alle lopende bewerkingen en plaatst de 80386 in een bekende gereset-te toestand.
- NC, No Connect
  Hier mag niets op aangesloten worden. Wanneer dit toch gebeurt kan de werking van de processor worden verstoord of is het onmogelijk om latere versies van de 80386 in dezelfde voet te steken.
- V_{CC}, System Power (aktief-HOGE ingang)
  Op deze pen wordt de +5 V nominale gelijkspanning van de systeemvoeding aangesloten.
- V_{SS}, System Ground (ingang)
  De systeem-aarde levert de 0 V verbinding, ten opzichte waarvan alle ingangs- en uitgangssignalen worden gemeten.

32 bit architectuur

De (80)386DX microprocessor bestaat uit een centrale verwerkingseenheid (CPU), een memory management unit (MMU) en een bus-interface. De centrale verwerkingseenheid bestaat uit een executie unit en een instructie unit. De executie unit bevat de acht algemene registers (general purpose registers) die worden gebruikt voor adresberekening, data-operaties en een 64 bit barrel-shifter. De instructie unit decodeert de instructie-opcodes en slaat die op in de gedecodeerde instructie-wachtrij voor onmiddellijk gebruik bij de executie unit. De memory management unit (MMU) bestaat uit een segmentatie unit en een paging unit. Door segmentatie kan de logische adresruimte worden beheerd doordat een extra adresseer-component wordt verschaft waarmee gemakkelijk code en data kan worden verplaatst. Het paging mechanisme werkt beneden het segmentatieproces en is daar ook transparant voor, om beheer van de fysieke adresruimte mogelijk te maken. Elk segment wordt verdeeld in één of meer 4 kB pagina's. Het geheugen is georganiseerd in één of meer segmenten met variabele lengte (elk tot maximaal 4 GB). Een bepaald gebied (segment) van de lineaire adresruimte kan vergezeld gaan van attributen, zoals plaats, afmeting, type en beveiligingskarakteristieken. Elke taak van een 386DX processor kan maximaal 16.381 segmenten van maximaal 4 GB per stuk hebben, zodat per taak 64 TB virtueel geheugen kan worden gebruikt.

De segmentatie unit levert vier beveiligingsniveaus om toepassingen en het bedrijfssysteem van elkaar te isoleren en tegen elkaar te beschermen.

Bedrijfsmodes

De 386DX heeft twee bedrijfsmodes: de Real Address Mode (real mode) en de Protected Virtual Address Mode (protected mode). In de real mode werkt de 386DX processor als een zeer snelle 8086, maar met 32 bit uitbreidingen als dat gewenst is. De real mode is voornamelijk nodig om de processor voor te bereiden op de protected mode. De protected mode geeft toegang tot het uitgekiende geheugenbeheer met paging en privileges.

Binnen de protected mode kan met behulp van software worden omgeschakeld naar virtuele 8086-mode taken.

4.1 80386

Om goede systeem-hardware ontwerpen mogelijk te maken is de bus-interface van de 386DX processor geschikt voor adres-pijplijning, dynamische bus-breedte en directe byte enable-signalen voor ieder byte van de databus.

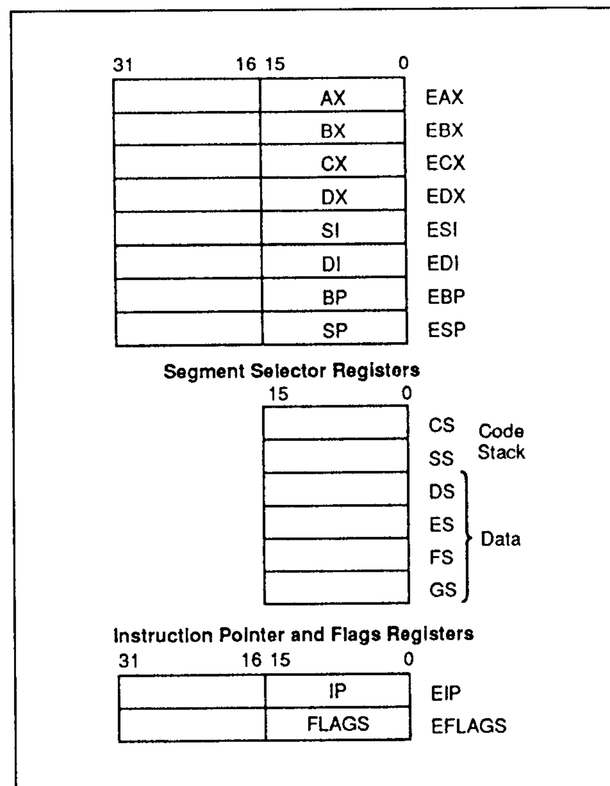
De registers

Overzicht van de registers

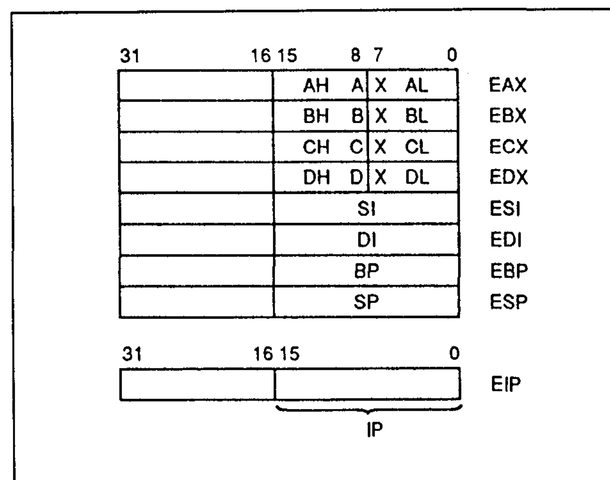
De 386DX kan beschikken over 32 registers in de volgende categorieën: algemene registers, segment-registers, instructie-pointer en flags, besturings-registers, systeem-adres registers, debug-registers en test-registers. Deze registers zijn een superset van de 8086, 80186 en 80286 registers (alle 16 bit registers van de deze processoren bevinden zich dus ook in de 32 bit 386DX microprocessor). In figuur 7/4.1-5 zijn alle registers te zien die met de basisarchitectuur te maken hebben. Hieronder vallen de algemene adres- en dataregisters, de instructie-pointer en het flags-register. De inhouden van deze registers zijn taak-specifiek, zodat deze registers bij een taakomschakeling automatisch met een nieuwe context worden geladen. De basisarchitectuur omvat ook zes direct toegankelijke segmenten die elk maximaal 4 GB groot kunnen zijn. Er wordt naar de segmenten verwezen door de selector-waarden die in de segment-registers van figuur 7/4.1-5 staan. Tijdens de uitvoering van een programma kunnen eventueel verschillende selector-waarden worden geladen. De selectors zijn ook taak-specifiek, dus worden de segment-registers ook automatisch geladen bij een taakomschakeling. De overige registers worden voornamelijk gebruikt door de systeem-software.

Algemene registers

De acht 32 bit algemene (General Purpose) registers bevatten data of adreswaarden. Deze registers (figuur 7/4.1-6) ondersteunen data-operands van 1, 8, 16, 32 en 64 bit en bit-velden van 1 tot 32 bit, terwijl adres-operands van 16 en 32 bit worden ondersteund. De 32 bit registers hebben de benamingen EAX, EBX, ECX, EDX, ESI, EDI, EBP en ESP.

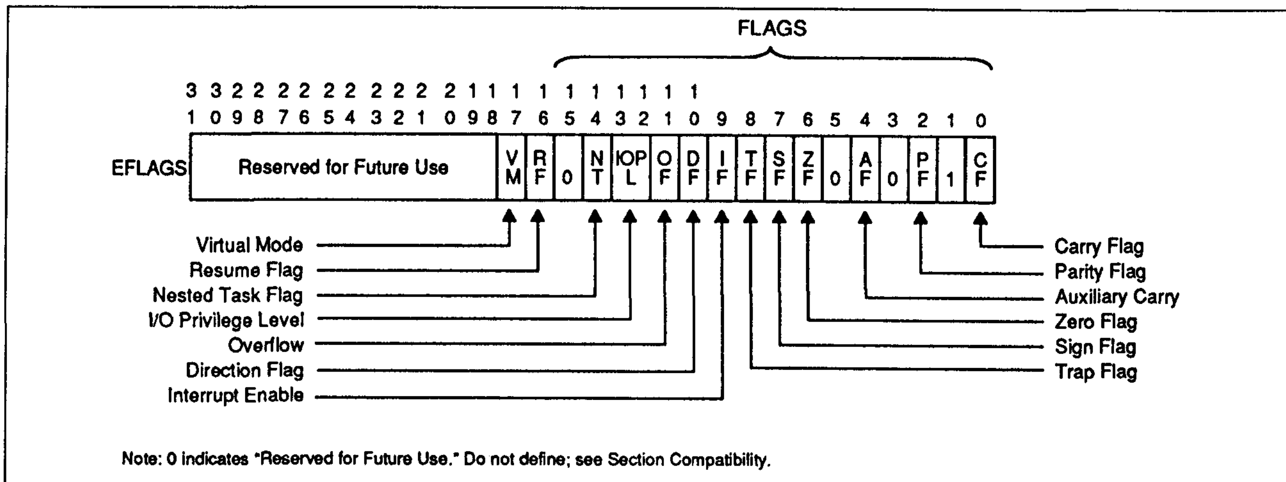


Figuur 7/4.1-5: De basisarchitectuur-registers van de 386DX.



Figuur 7/4.1-6: De algemene registers en de instructie-pointer.

4.1 80386



Figuur 7/4.1-7: Het flags-register.

Toegang tot de laagste 16 bits van de registers kan apart worden verkregen door gebruik te maken van de 16 bit namen AX, BX, CX, DX, SI, DI, BP en SP. Wanneer ze als 16 bit operand worden bereikt, worden de hoogste 16 bits van het register niet gebruikt en ook niet veranderd. Tenslotte kunnen 8 bit operaties individueel toegang krijgen tot het laagste byte (bits 0 - 7) en het daarop volgende byte (bits 8 - 15) van de algemene registers AX, BX, CX en DX. De laagste bytes heten AL, BL, CL en DL en de hogere bytes AH, BH, CH en DH.

De instructie-pointer

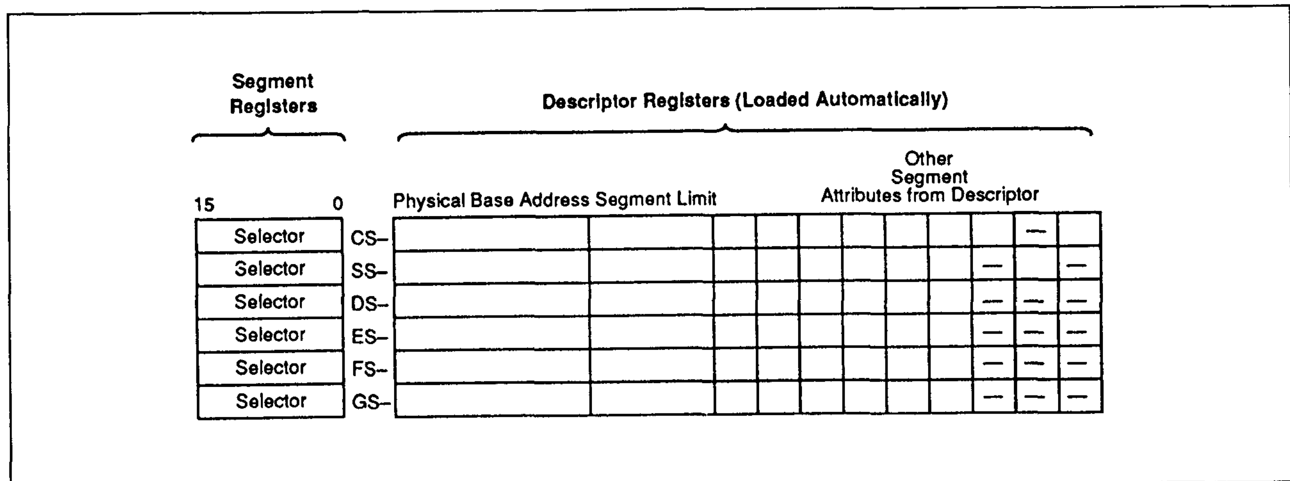
De instructie-pointer (EIP) is een 32 bit register die de offset van de daarna uit te voeren instructie bevat. De offset is altijd relatief ten opzichte van de basis van het code-segment (CS). De laagste 16 bits (bits 0 - 15) van EIP bevatten de 16 bit instructie-pointer (IP) die bij 16 bit adresseringen gebruikt wordt.

Flags-register

Ook het flags-register (EFLAGS) is 32 bit breed (zie figuur 7/4.1-7). De gedefinieerde bits en bit-velden in EFLAGS besturen bepaalde operaties en geven de status van de 386DX processor aan. De laagste 16 bits (bits 0 - 15) vormen het 16 bit flag-register (FLAGS) dat vooral bij de uitvoering van 8086 en 80286 code wordt gebruikt.

- VM
  Door de virtual 8086 modebit VM = 1 te maken (in de protected mode) komt de 386DX in de virtuele 8086 mode.
- RF
  De resume flag RF wordt samen met debug register breakpoints gebruikt.
- NT
  Het nested taskbit TF wordt op 1 gezet om aan te geven dat de uitvoering van de lopende taak is genesteld binnen een andere taak.
- IOPL
  Met dit tweebits veld wordt het numerieke maximum CPL (current privilege level) aangegeven dat mag worden gebruikt zonder dat daardoor een uitzondering 13 fout wordt veroorzaakt.
- OF
  De overflow flag wordt gezet als de operatie leidde tot een overflow met teken.
- DF
  De direction flag bepaalt of ESI en/of EDI registers tijdens de string-operaties post-incrementeren (DF = 1) of post-decrementeren.

4.1 80386

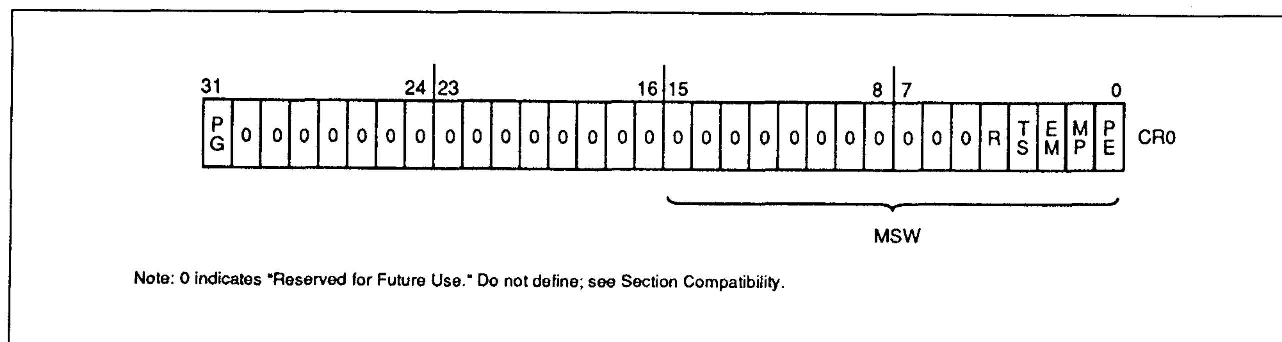


Figuur 7/4.1-8: Segment-registers en de bijbehorende descriptor-registers.

- IF
  Als de interrupt enable flag is gezet, worden externe interrupts op de INTR-pen herkend.
- TF
  Wanneer de trap enable flag TF = 1 is, genereert de 386DX een uitzondering 1 trap na uitvoering van de volgende instructie.
- SF
  Het signbit wordt gezet als het hoogste bit van het resultaat 1 is. Voor 8, 16 en 32 bit operaties geeft SF de toestand van bit 7, 15 of 31 weer.
- ZF
  De zero flag is 1 als alle bits van het resultaat 0 zijn.
- AF
  De auxiliary flag wordt gebruikt om het optellen en aftrekken van gepakte BCD-grootheden te vergemakkelijken (AF = 1 bij een carry uit of een borrow naar bit 3).
- PF
  De parity flag wordt gezet als de laagste 8 bits van de operatie een even aantal enen bevat.
- CF
  De carry flag wordt gezet als de operatie resulteert in een carry uit of een borrow naar het hoogste bit.

Segment-registers
In zes 16 bit segment-registers (figuur 7/4.1-8) worden de segment-selectorwaarden opgeslagen die de actuele adresseerbare geheugen-segmenten identificeren. In de beveiligde (protected) mode kan elk segment een grootte hebben van één byte tot de gehele lineaire en fysieke ruimte van de machine (2^{32} bytes). In de Real Address Mode bedraagt de maximale segmentgrootte 64 kB (2^{16} bytes). De zes segmenten die op ieder willekeurig moment adresseerbaar zijn, worden gedefinieerd door de segment-registers CS, SS, DS, ES, FS en GS. De selector in CS geeft het actuele code-segment weer; de selector in SS laat het actuele stack-segment zien en de selectors in DS, ES, FS en GS wijzen op de actuele data-segmenten.

Segment descriptor-registers
De segment descriptor-registers zijn voor de programmeur niet zichtbaar, maar het is wel nuttig om de inhoud ervan te begrijpen. Binnen de 386DX hoort elk descriptor-register bij een (zichtbaar) segment-register.
Elk descriptor-register bevat een 32 bit segmentbasisadres, een 32 bit segmentlimiet en de overige noodzakelijke segment-attributen.



Figuur 7/4.1-9: Control Register 0 (CR0).

Wanneer een selectorwaarde in een segment-register wordt geladen, wordt het bijbehorende descriptor-register automatisch voorzien van de juiste informatie. In de Real Address Mode wordt alleen het basisadres direct aangepast (door de selectorwaarde vier bits naar links te schuiven), aangezien de maximum segment-limiet en de attributen in de real mode zijn gefixeerd. In de Protected Mode worden het basisadres, de limiet en de attributen allemaal bijgewerkt door de inhoud van de segment-descriptor die wordt aangewezen door de selector. Wanneer naar geheugen wordt verwezen, wordt het 32 bit segment-basisadres een component van de lineaire adresberekening.
De 32 bit limiet wordt dan gebruikt voor de limiet-check operatie en de attributen worden aan de hand van de soort geheugenverwijzing gecheckt.

Control-registers

De 386DX heeft drie 32 bit besturingsregisters (CR0, CR2 en CR3) die algemene machine-informatie bevatten (niet specifiek voor een bepaalde taak).

CR0, Machine Control Register (bevat ook het 80286 Machine Statuswoord)

CR0 (figuur 7/4.1-9) bevat zes bits voor besturings- en statusdoeleinden. De laagste 16 bits worden ook wel het Machine Status Word (MSW) genoemd voor compatibiliteit met de 80286 Protected Mode.

- PG
  Het paging enable-bit wordt 1 om de op de chip aanwezige paging-unit vrij te geven.
- R
  R is gereserveerd voor toekomstig gebruik.
- TS
  De task switch wordt automatisch gezet als een taakomschakeling heeft plaatsgevonden.
- EM
  Het emulate coprocessor-bit wordt gezet om alle coprocessor opcodes een Coprocessor Not Available fout (uitzondering 7) te laten veroorzaken.
- MP
  Het monitor coprocessor-bit wordt samen met het TS-bit gebruikt om te bepalen of de WAIT opcode een Coprocessor Not Available fout genereert. Als MP en TS beide 1 zijn, is dat het geval.
- PE
  Het protection enable-bit wordt gezet om de Protected Mode vrij te geven.

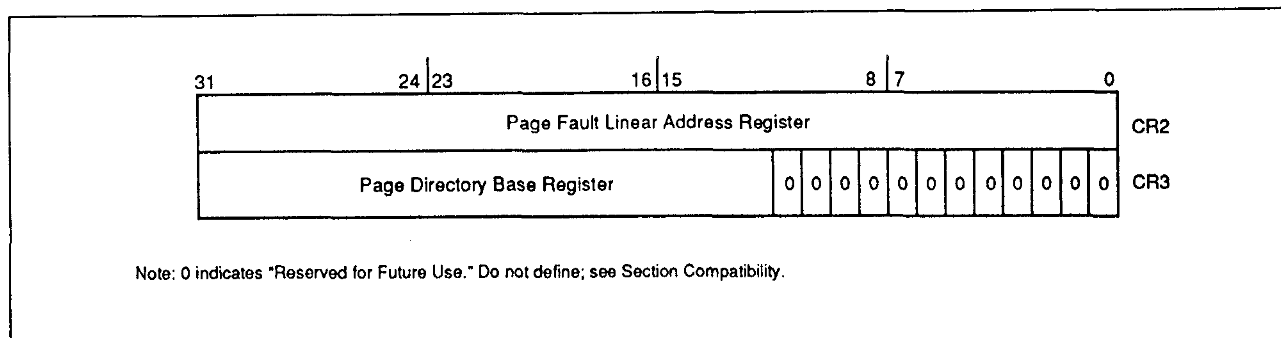
CR1

Gereserveerd.

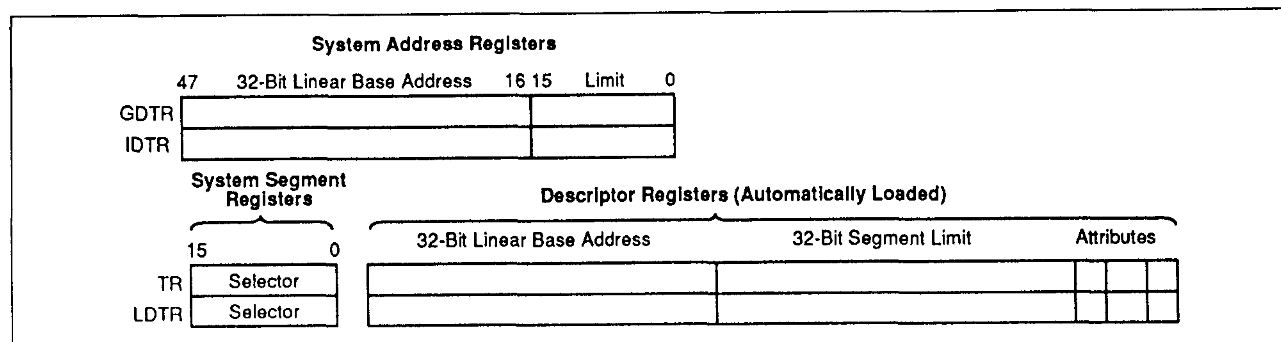
CR2: Page Fault Linear Address

CR2 (figuur 7/4.1-10) bevat het 32 bit lineaire adres dat de laatst gedetecteerde paginafout veroorzaakte. De foutcode die op de foutbehandelings-stack werd gezet, levert extra informatie over deze fout.

4.1 80386



Figuur 7/4.1-10: De besturingsregisters CR2 en CR3.



Figuur 7/4.1-11: De systeem-adresregisters en de systeem-segmentregisters.

CR3: Page Directory Base Address

CR3 (ook te zien in figuur 7/4.1-10) bevat het fysieke basisadres van de pagina-directory tabel die bij de 386DX altijd pagina-uitgericht is (4 kB uitrichting). Daarom worden de laagste 12 bits van CR3 genegeerd.

Systeem-adresregisters

Voor de verwijzing naar de tabellen of segmenten die door de 80286 CPU en het 386DX microprocessor beveiligingsmodel worden ondersteund staan vier speciale registers ter beschikking. Deze tabellen of segmenten zijn: GDT (Global Descriptor Table), IDT (Interrupt Descriptor Table), LDT (Local Descriptor Table) en TSS (Task State Segment).

De adressen van deze tabellen of segmenten worden in speciale registers opgeslagen: de System Address Registers en de System Segment Registers (zie figuur 7/4.1-11). De namen hiervan zijn GDTR, IDTR, LDTR en TR.

GDTR en IDTR

Deze registers bevatten het 32 bit lineaire basisadres van de GDT en de 16 bit limiet van de IDT. Aangezien de GDT en IDT segmenten globaal zijn voor alle taken in het systeem, worden zij gedefinieerd door 32 bit lineaire adressen en 16 bit limietwaarden.

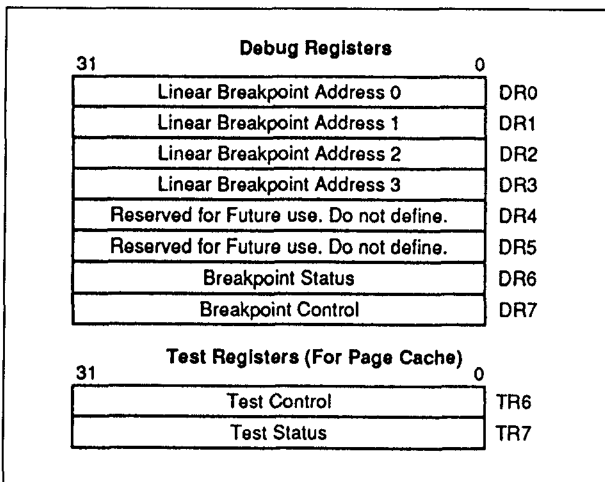
LDTR en TR

Deze registers bevatten achtereenvolgens de 16 bit selector voor de LDT descriptor en de TSS descriptor. Aangezien de LDT en TSS segmenten taak-specifieke segmenten zijn, worden zij gedefinieerd door selectorwaarden die in de systeem segmentregisters zijn opgeslagen.

Debug Registers

De zes voor de programmeur toegankelijke debug registers ondersteunen het foutzoeken op de chip. De Debug Registers DR0 tot en met 3 specificeren vier lineaire breekpunten. Het Debug Control Register DR7 wordt

gebruikt om de breekpunten in te stellen en het Debug Status Register DR6 laat de huidige toestand ervan zien.



Figuur 7/4.1-12: De Debug en Test Registers.

Test Registers
Voor de besturing van het testen van de RAM/CAM (Content Adressable Memories) in het Translation Lookaside Buffer-gedeelte van de 386DX processor worden twee registers gebruikt. TR6 is het commando-testregister en TR7 is het register dat de data van de Translation Lookaside Buffer-test bevat.

De toegankelijkheid van de registers
De manieren waarop de registers worden bereikt vertonen wat verschillen voor de Real Mode en de Protected Mode. Deze verschillen worden in tabel 7/4.1-3 samengevat.

De instructieset

Negen categorieën
De instructie-set van de 386DX beslaat negen categorieën:
- data-overdracht;
- rekenkundige instructies;
- shift/rotate;
- string-manipulaties;
- bit-manipulaties;
- besturings-overdracht;
- ondersteuning van hogere talen;
- ondersteuning van bedrijfssystemen;
- processor besturing.

Zie voor een overzicht tabel 7/4.1-4.

Operands
Alle instructies werken met 0, 1, 2 of 3 operands, waarbij een operand zich in een register, in de instructie zelf of in geheugen kan bevinden. De meeste nul-operand instructies (zoals CLI, STI) zijn slechts één byte groot. Eén-operand instructies zijn meestal twee bytes lang. De gemiddelde instructie heeft een lengte van 3,2 bytes.

Register	Use In Real Mode		Use In Protected Mode		Use In Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
General Registers	Yes	Yes	Yes	Yes	Yes	Yes
Segment Registers	Yes	Yes	Yes	Yes	Yes	Yes
Flag Registers	Yes	Yes	Yes	Yes	IOPL	IOPL
Control Registers	Yes	Yes	PL = 0	PL = 0	No	Yes
GDTR	Yes	Yes	PL = 0	Yes	No	Yes
IDTR	Yes	Yes	PL = 0	Yes	No	Yes
LDTR	No	No	PL = 0	Yes	No	No
TR	No	No	PL = 0	Yes	No	No
Debug Control	Yes	Yes	PL = 0	PL = 0	No	No
Test Registers	Yes	Yes	PL = 0	PL = 0	No	No

Notes: PL = 0: The registers can be accessed only when the current privilege level is zero.
IOPL: The PUSHF and POPF instructions are made I/O Privilege Level sensitive in Virtual 8086 Mode.

Tabel 7/4.1-3: Het gebruik van de registers bij verschillende bedrijfsmodes.

4.1 80386

General Purpose	
MOV	Move operand
PUSH	Push operand onto stack
POP	Pop operand off stack
PUSHA	Push all registers on stack
POPA	Pop all registers off stack
XCHG	Exchange operand register
XLAT	Translate
Conversion	
MOVZX	Move byte or Word, Dword with zero extension
MOVSX	Move byte or Word, Dword, sign extended
CBW	Convert byte to Word, or Word to Dword
CWD	Convert Word to Dword
CWDE	Convert Word to Dword extended
CDQ	Convert Dword to Qword
Input/Output	
IN	Input operand from I/O space
OUT	Output operand to I/O space
Address Object	
LEA	Load effective address
LDS	Load pointer into D segment register
LES	Load pointer into E segment register
LFS	Load pointer into F segment register
LGS	Load pointer into G segment register
LSS	Load pointer into S (Stack) segment register
Flag Manipulation	
LAHF	Load A register from Flags
SAHF	Store A register in Flags
PUSHF	Push flags onto stack
POPF	Pop flags off stack
PUSHFD	Push EFLAGS onto stack
POPFD	Pop EFLAGS off stack
CLC	Clear Carry Flag
CLD	Clear Direction Flag
CMC	Complement Carry Flag
STC	Set Carry Flag
STD	Set Direction Flag

Tabel 7/4.1-4a: Overzicht van de 80386 instructies voor data-overdracht.

Aangezien de 386DX een 16 byte instructiewachtrij heeft, worden gemiddeld 5 instructies van tevoren opgehaald (ge"prefetched"). Door twee operands te gebruiken zijn de volgende gemeenschappelijke instructies mogelijk:
- register-naar-register;
- geheugen-naar-register;
- immediate-naar-register;
- register-naar-geheugen;
- immediate-naar-geheugen.

Addition	
ADD	Add operands
ADC	Add with carry
INC	Increment operand by 1
AAA	ASCII adjust for addition
DAA	Decimal adjust for addition
Subtraction	
SUB	Subtract operands
SBB	Subtract with borrow
DEC	Decrement operand by 1
NEG	Negate operand
CMP	Compare operands
DAS	Decimal adjust for subtraction
AAS	ASCII adjust for subtraction
Multiplication	
MUL	Multiply Double/Single Precision
IMUL	Integer multiply
AAM	ASCII adjust after multiply
Division	
DIV	Divide unsigned
IDIV	Integer divide
AAD	ASCII adjust before division

Tabel 7/4.1-4b: Overzicht van de 80386 instructies voor rekenkundige instructies.

MOVS	Move byte or Word, Dword string
INS	Input string from I/O space
OUTS	Output string to I/O space
CMPS	Compare byte or Word, Dword string
SCAS	Scan Byte or Word, Dword string
LODS	Load byte or Word, Dword string
STOS	Store byte or Word, Dword string
REP	Repeat
REPE/ REPZ	Repeat while equal/zero
RENE/ REPNZ	Repeat while not equal/not zero

Tabel 7/4.1-4c: Overzicht van de 80386 instructies voor logische instructies.

4.1 80386

Logicals	
NOT	"NOT" operand
AND	"AND" operands
OR	"Inclusive OR" operands
XOR	"Exclusive OR" operands
TEST	"Test" operands
Shifts	
SHL/SHR	Shift logical left or right
SAL/SAR	Shift arithmetic left or right
SHLD/ SHRD	Double shift left or right
Rotates	
ROL/ROR	Rotate left/right
RCL/RCR	Rotate through carry left/right

Tabel 7/4.1-4d: Overzicht van de 80386 instructies voor string instructies.

Single Bit Instructions	
BT	Bit Test
BTS	Bit Test and Set
BTR	Bit Test and Reset
BTC	Bit Test and Complement
BSF	Bit Scan Forward
BSR	Bit Scan Reverse

Tabel 7/4.1-4e: Overzicht van de 80386 instructies voor programma besturings instructies.

De operands kunnen 8, 16 of 32 bits lang zijn. Over het algemeen geldt dat bij het uitvoeren van code die voor de 386DX is geschreven (32 bit code) de operands 8 of 32 bit zijn. Wordt bestaande 80286 of 8086 code uitgevoerd (16 bit code) dan zijn de operands 8 of 16 bit. Aan alle instructies kunnen prefixes worden toegevoegd waardoor de default-lengte van de operands wordt vervangen door nieuwe (er worden dan 32 bit operands gebruikt voor 16 bit code of 16 bit operands voor 32 bit code).

Conditional Transfers	
SETCC	Set byte equal to condition code
JA/JNBE	Jump if above/not below nor equal
JAE/JNB	Jump if above or equal/not below
JB/JNAE	Jump if below/not above nor equal
JBE/JNA	Jump if below or equal/not above
JC	Jump if carry
JE/JZ	Jump if equal/zero
JG/JNLE	Jump if greater/not less nor equal
JGE/JNL	Jump if greater or equal/not less
JL/JNGE	Jump if less/not greater nor equal
JLE/JNG	Jump if less or equal/not greater
JNC	Jump if not carry
JNE/JNZ	Jump if not equal/not zero
JNO	Jump if not overflow
JNP/JPO	Jump if not parity/parity odd
JNS	Jump if not sign
JO	Jump if overflow
JP/JPE	Jump if parity/parity even
JS	Jump if sign

Tabel 7/4.1-4f: Overzicht van de 80386 instructies voor bit-manipulatie instructies (deel 1).

Adresseringsmodes

De 386DX processor biedt in totaal elf adresseringsmodes voor instructies om de operands te specificeren. Hiervan zijn er twee waarbij de instructies op register operands of immediate operands werken. Bij de Register Operand Mode bevindt de operand zich in één van de 8, 16 of 32 bit algemene registers. Bij de Immediate Operand Mode maakt de operand als gedeelte van de opcode deel uit van de instructie. De resterende 9 modes leveren een mechanisme op om het effectieve adres van een operand te specificeren. Het lineaire adres bestaat uit twee componenten: het segment-basisadres en een effectief adres. Het effectieve adres (EA) wordt met de volgende formule berekend:
EA = basis + (index * schaal) + verplaatsing
De basis- en index-variabelen zijn waarden uit algemene registers, terwijl de waarde van

de verplaatsing (displacement) in de instructie is opgenomen. Elk algemeen register kan dienen als basis- of index-register. De waarde in het index-register kan worden geschaald (vermenigvuldigd met 1, 2, 4 of 8). Een verplaatsingswaarde kan 8 of 32 bits lang zijn en wordt door de processor geïnterpreteerd als een 2's complement waarde met teken. In figuur 7/4.1-13 zijn de gevolgen van de adresberekeningen te zien.

Unconditional Transfers	
CALL	Call procedure/task
RET	Return from procedure
JMP	Jump
Iteration Controls	
LOOP	Loop
LOOPE/ LOOPZ	Loop if equal/zero
LOOPNE/ LOOPNZ	Loop if not equal/not zero
JCXZ	JUMP if register CX = 0
Interrupts	
INT	Interrupt
INTO	Interrupt if overflow
IRET	Return from interrupt/task
CLI	Clear interrupt enable
STI	Set interrupt enable

Tabel 7/4.1-4g: Overzicht van de 80386 instructies voor bit-manipulatie instructies (deel 2).

BOUND	Check array bounds
ENTER	Setup parameter block for entering procedure
LEAVE	Leave procedure

Tabel 7/4.1-4h: Overzicht van de 80386 instructies voor ondersteuning van hogere talen.

SGDT	Store global descriptor table
SIDT	Store interrupt descriptor table
STR	Store task register
SLDT	Store local descriptor table
LGDT	Load global descriptor table
LIDT	Load interrupt descriptor table
LTR	Load task register
LLDT	Load local descriptor table
ARPL	Adjust requested privilege level
LAR	Load access rights
LSL	Load segment limit
VERR/ VERW	Verify segment for reading or writing
LMSW	Load machine status word (lower 16 bits of CR0)
SMSW	Store machine status word

Tabel 7/4.1-4i: Overzicht van de 80386 instructies voor ondersteuning van bedrijfssystemen.

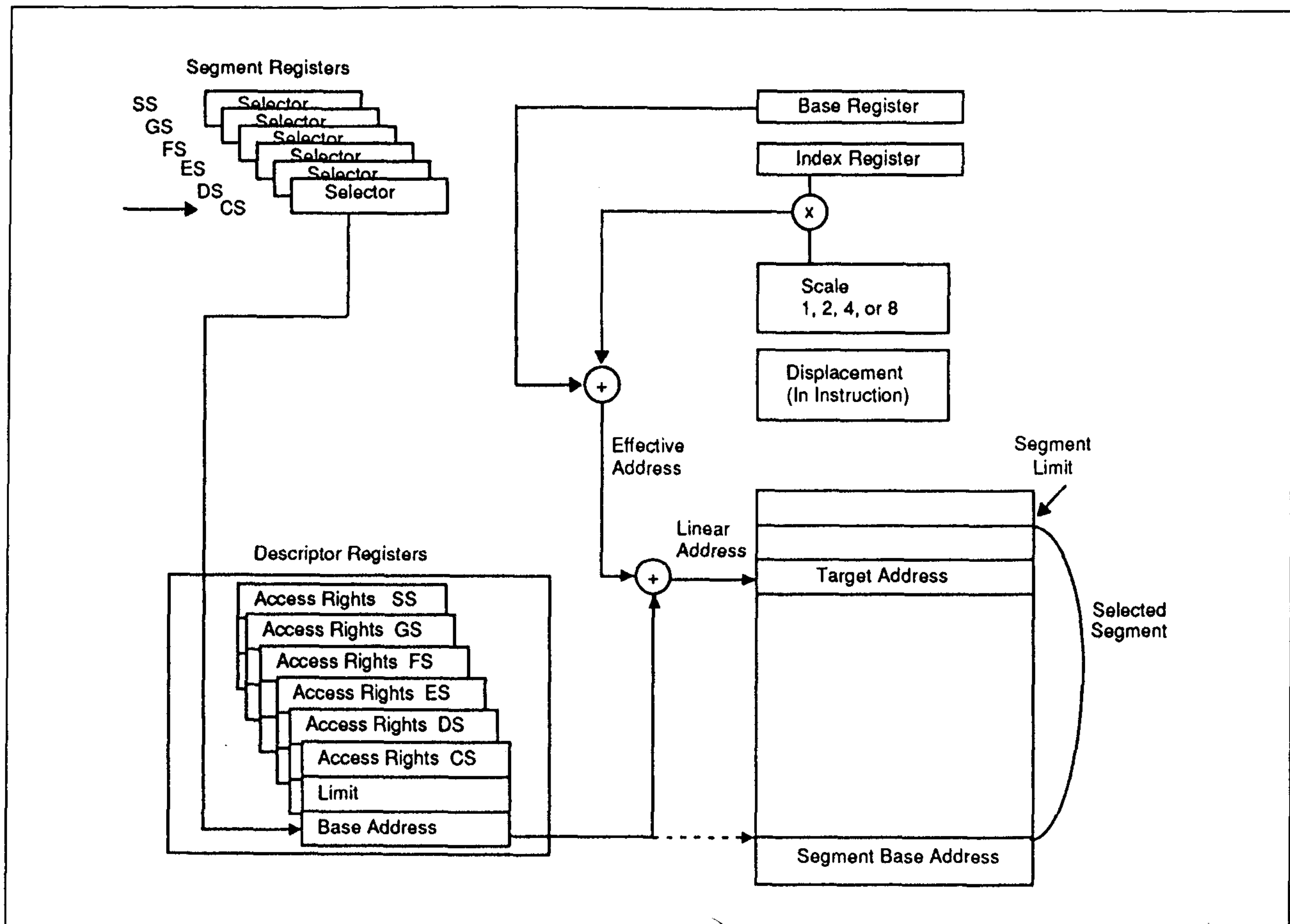
HLT	Halt
WAIT	Wait until $\overline{BUSY}$ negated
ESC	Escape
LOCK	Lock Bus

Tabel 7/4.1-4j: Overzicht van de 80386 instructies voor processor besturing.

De 9 resterende adresserings-modes worden gevormd door combinaties van de 4 bovenstaande componenten:

- Direct Mode
  De offset van de operand maakt deel uit van de instructie als 8, 16 of 32 bit verplaatsing. Bijvoorbeeld: INC Word PTR [500].
- Register Indirect Mode
  Het adres van de operand staat in een basis-register.
  Bijvoorbeeld: MOV [ECX],EDX.
- Based Mode

4.1 80386



Figuur 7/4.1-13: Berekening van de adressen.

- De inhoud van een basis-register wordt opgeteld bij een verplaatsing. Bijvoorbeeld: MOV ECX,[EAX + 24].
- Index Mode
  De inhoud van een index-register wordt opgeteld bij een verplaatsing. Bijvoorbeeld: ADD EAX, TABLE[ESI].
- Scaled Index Mode
  De inhoud van een index-register wordt vermenigvuldigd met een schaalfactor en opgeteld bij een verplaatsing.
  Bijvoorbeeld:
  IMUL EBX, TABLE[ESI*4],7.
- Based Index Mode
  De inhoud van een basis-register wordt opgeteld bij de inhoud van een index-register.
  Bijvoorbeeld:
  MOV EAX, [ESI] [EBX].
- Based Scaled Index Mode
  De inhoud van een index-register wordt vermenigvuldigd met een schaalfactor en opgeteld bij de inhoud van een basis-register. Bijvoorbeeld:
  MOV ECX, [EDX*8] [EAX].
- Based Index Mode with Displacement
  De inhoud van een index-register wordt opgeteld bij de inhoud van een basis-register en een verplaatsing. Bijvoorbeeld:
  ADD EDX, [ESI] [EBP + 00FFFFF0H].
- Based Scaled Index Mode with Displacement
  De inhoud van een index-register wordt vermenigvuldigd met een schaalfactor en opgeteld bij de inhoud van een basis-register en een verplaatsing. Bijvoorbeeld:

4.1 80386

MOV EAX, LOCALTABLE[EDI*4] [EBP + 80].

Verschillen tussen 16 en 32 bit adressen
Om de software geschikt te houden voor zowel de 8086, de 80286 als de 80386, kan de 386DX 16 bit instructies in Real en Protected Mode uitvoeren. De processor bepaalt de grootte van de instructies die hij uitvoert door het D-bit in de CS segmentdescriptor te bekijken. Als het D-bit = 0 is, wordt aangenomen dat alle operand-lengten en effectieve adressen 16 bit lang zijn. Als het D-bit = 1 is, is de default-lengte van de operands en de adressen 32 bit. In de Real Mode is de default-waarde van operands en adressen 16 bit.
Zonder te letten op de default precisie van de operands of adressen is de 386DX in staat om zowel 16 bit als 32 bit instructies uit te voeren. Dit wordt gespecificeerd door middel van override prefixes. Er zijn twee prefixes, de "Operand Size Prefix" en de "Address Length Prefix", die de waarde van het D-bit op individuele basis overschrijven.

- Voorbeeld 1
  De processor is bezig in Real Mode en de programmeur moet toegang krijgen tot de EAX-registers. De assembler-code hiervoor zou kunnen zijn: MOV EAX, 32 bit MEMORYOP. Een assembler zoals de ASM386 bepaalt dan automatisch dat een Operand Size Prefix nodig is en genereert die.
- Voorbeeld 2
  Met D = 0 wil de programmeur Scaled Index adressering gebruiken om toegang te krijgen tot een array. De Address Length Prefix staat het gebruik van MOV DX, TABLE[ESI*2] toe. De assembler gebruikt een Address Length Prefix omdat, met D = 0, de default adresseringsmode 16 bit is.

Beide genoemde prefixes kunnen apart of in combinatie op elke instructie worden toegepast. Bij de Address Length Prefix is het niet toegestaan in Real Mode naar adressen groter dan 64 kB te gaan. Een geheugenadres van meer dan FFFFH resulteert in een algemene beveiligingsfout. Een Address Length Prefix maakt alleen toepassing van de extra 386DX adresseringsmodes mogelijk. Wanneer de 386DX bezig is met 32 bit code, zijn de verplaatsingen 8 of 32 bit en kan elk willekeurig register als basis- of index-register worden gebruikt. Bij het uitvoeren van 16 bit code zijn de verplaatsingen 8 of 16 bit en de basis- en indexregisters overeenkomstig het 80286 model (zie tabel 7/4.1-5).

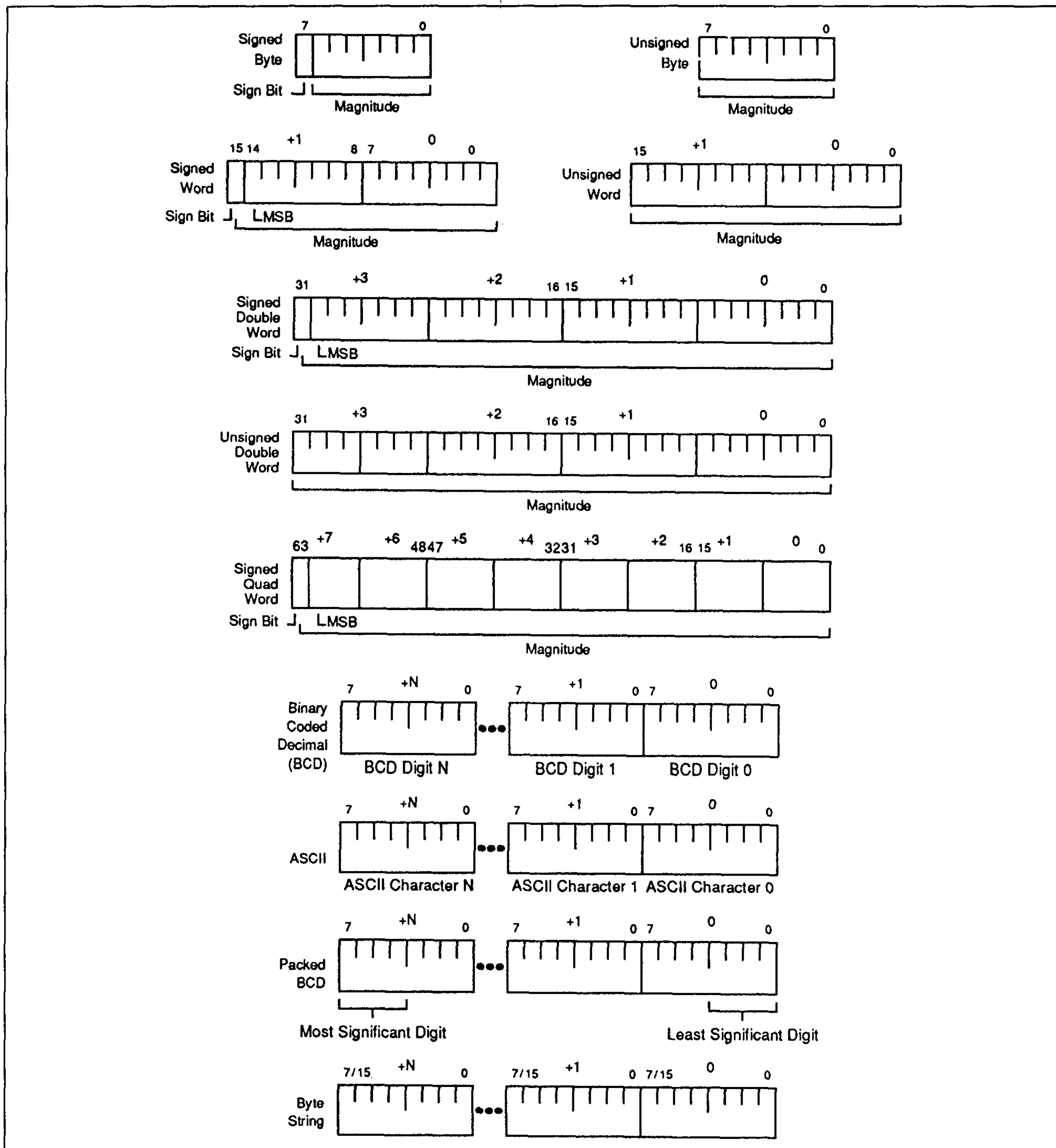
	16-Bit Addressing	32-Bit Addressing
Base Register	BX, BP	Any 32-bit GP Register
Index Register	SI, DI	Any 32-bit GP Register Except ESP
Scale Factor	None	1, 2, 4, 8
Displacement	0, 8, 16 bits	0, 8, 32 bits

Tabel 7/4.1-5: De basis- en index-registers voor 16 en 32 bit adressen.

Data-typen
De 386DX microprocessor ondersteunt alle data-typen die gewoonlijk in hogere talen worden gebruikt (zie ook figuur 7/4.1-14):

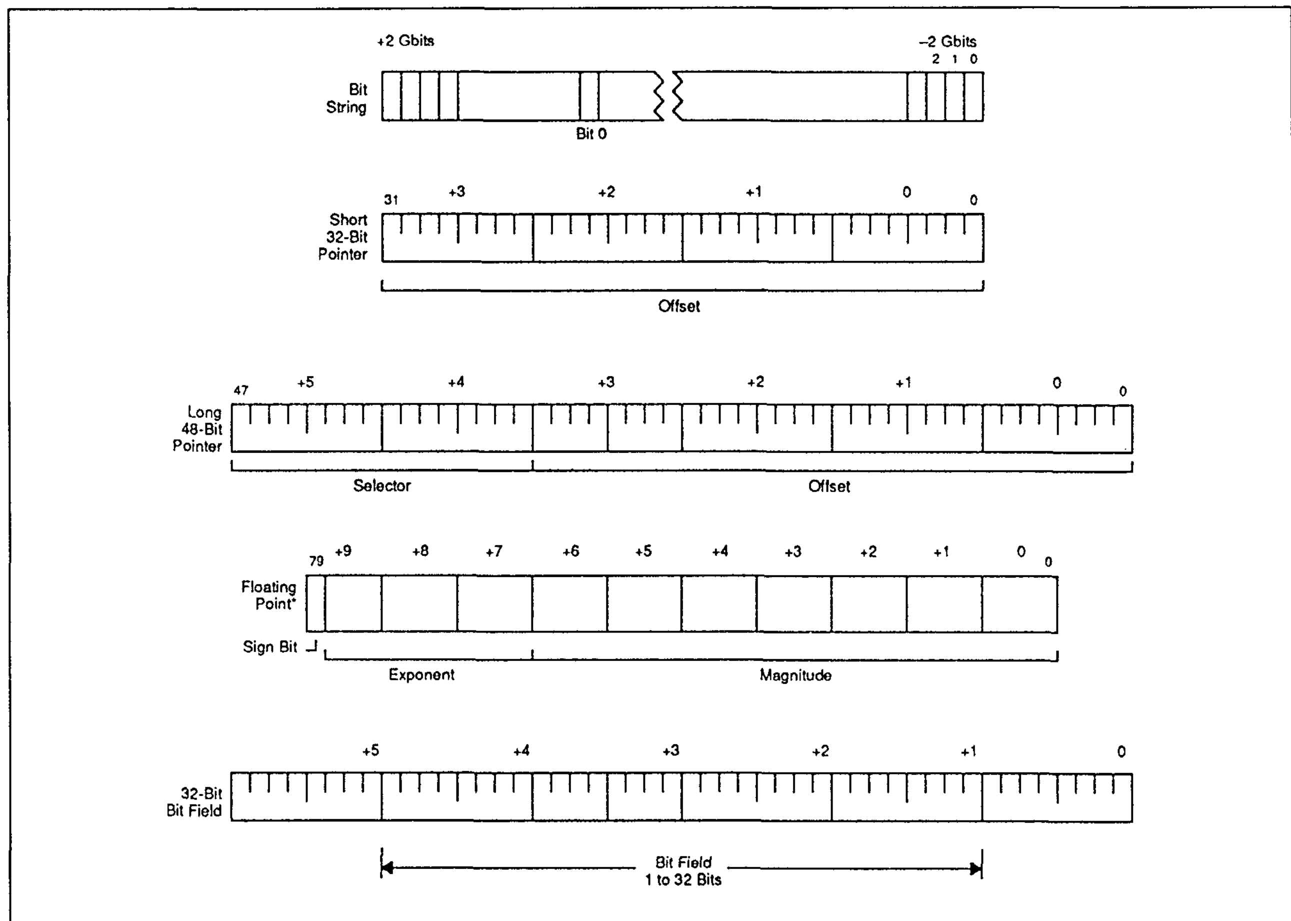
- Bit:
  een waarde ter grootte van een enkel bit.
- Bit Field:
  een groep van maximaal 32 aangrenzende bits, verdeeld over maximaal 4 bytes.
- Bit String:
  bij de 386DX mag een set aangrenzende bits maximaal 4 GB lang zijn.
- Byte:
  een hoeveelheid van 8 bit, inclusief teken.
- Unsigned Byte:
  een omvang van 8 bit, zonder teken.
- Integer (Word):
  een grootte van 16 bit, inclusief teken.
- Long Integer (Double Word):
  een omvang van 32 bit, inclusief teken. Bij alle operaties wordt een 2's complement representatie verondersteld.
- Unsigned Integer (Word):
  een 16 bit grootte, zonder teken.
- Unsigned Long Integer (Double Word):
  een hoeveelheid van 32 bit, zonder teken.

4.1 80386



Figuur 7/4.1-14a: Alle door de 386DX ondersteunde data-typen (de Floating Point alleen door de 387DX numerieke coprocessor), deel 1.

4.1 80386



Figuur 7/4.1-14b: Alle door de 386DX ondersteunde data-typen (de Floating Point alleen door de 387DX numerieke coprocessor), deel 2.

- Signed Quad Word:
  een 64 bit waarde, inclusief teken.
- Unsigned Quad Word:
  een 64 bit waarde, zonder teken.
- Offset:
  een 16 of 32 bit offset-waarde die indirect naar een andere geheugenlokatie verwijst.
- Pointer:
  een volledige pointer bestaat uit een 16 bit segment-selector en 16 of 32 bit offset.
- Char:
  een byte-representatie van een ASCII alfanumeriek of besturings-karakter.
- String:
  een aangrenzende volgorde van bytes, words of dwords.
  Een string kan tussen 1 byte en 4 GB groot zijn.
- BCD:
  een (ongepakte) byte-representatie van een decimaal cijfer (0 - 9).
- Packed BCD:
  een (gepakte) voorstelling van twee decimale cijfers 0 - 9, met één cijfer in elke nibble.

386DX/387DX combinatie

Wanneer de 386DX is gekoppeld aan een 387DX Numerieke Coprocessor, worden de volgende drijvende komma (floating point) typen ondersteund:

- Floating Point:
  de representatie van een 32, 64 of 80 bit geheel getal, inclusief teken.

4.1 80386

Geheugen-organisatie

Inleiding

Het geheugen bij de 386DX processor is onderverdeeld in 8 bit (bytes), 16 bit (woorden) en 32 bit grootten (dubbel-woorden). Woorden worden opgeborgen in twee aaneengesloten bytes, met het laagste byte op het laagste adres. Dubbelwoorden (Dwords) worden op dezelfde manier opgeslagen in vier opeenvolgende bytes, met het laagste byte op het laagste adres en het hoogste byte op het hoogste adres. Het adres van een word of dword is dan het byte-adres van het laagste byte.
Bovendien ondersteunt de 386DX nog twee grotere geheugen-eenheden: pagina's en segmenten. Het geheugen kan worden onderverdeeld in één of meer segmenten met variabele lengte, waarvan meerdere programma's gebruik kunnen maken of die op schijf gezet kunnen worden. Het geheugen kan ook worden georganiseerd in één of meer 4 kB pagina's. Tenslotte kan zowel segmentatie als paginering worden gecombineerd om gebruik te maken van de voordelen van beide systemen.

Adresruimten

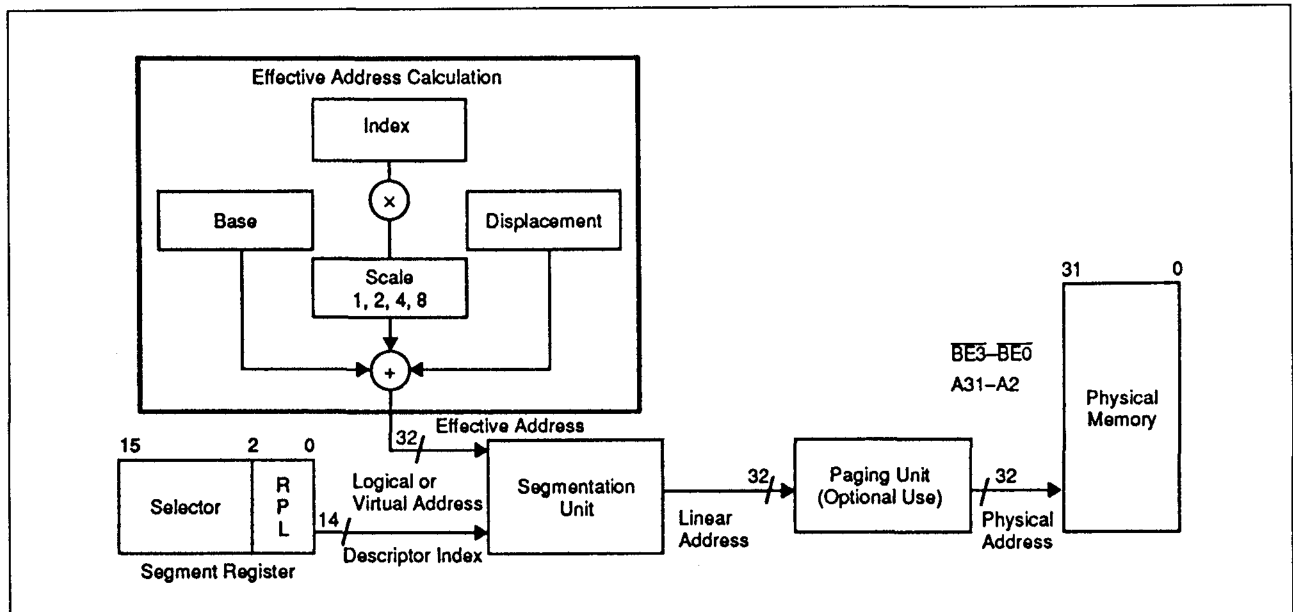
De 386DX heeft drie aparte adresruimten:
- logisch;
- lineair
- fysiek;

zie figuur 7/4.1-15. Een logisch adres (ook wel virtueel adres genoemd) bestaat uit een selector en een offset. Een selector is de inhoud van een segment-register. Een offset wordt gevormd door alle adresserings-componenten bij elkaar op te tellen (BASE, INDEX, DISPLACEMENT). Aangezien iedere taak bij een 386DX processor maximaal 16 K (2^{14} -1) selectors heeft en offsets 4 GB (2^{32} bits) groot kunnen zijn, bedraagt de logische adresruimte per taak maximaal 2^{46} bits (= 64 TB).
De segmentatie-eenheid vertaalt de logische adresruimte in een 32 bit lineaire adresruimte. Als de paging unit niet is vrijgegeven komt het 32 bit lineaire adres overeen met het fysieke adres. De paging unit vertaalt de lineaire adresruimte in fysieke adresruimte. Het fysieke adres is wat op de adrespennen verschijnt.
Het belangrijkste verschil tussen Real Mode en Protected Mode is hoe de segmentatie-eenheid de vertaling van logisch adres naar lineair adres uitvoert. In de Real Mode verschuift de segmentatie-eenheid de selector vier bits naar links en telt het resultaat op bij de offset om zo het lineaire adres te vormen. In de Protected Mode heeft elke selector een bijbehorend lineair basisadres. Het lineaire basisadres wordt opgeslagen in één of twee bedrijfssysteem-tabellen (bijvoorbeeld de Local Descriptor Table of de Global Descriptor Table). Het lineaire basisadres van de selector wordt opgeteld bij de offset om zodoende het uiteindelijke lineaire adres te vormen.

Het gebruik van de segment-registers

Het segment is de belangrijkste datastructuur voor het organiseren van geheugen. Bij de 386DX zijn segmenten blokken van lineaire adressen met verschillende afmetingen, waarbij bepaalde attributen behoren. Twee typen segmenten zijn het belangrijkst: code en data, die van 1 byte tot 4 GB groot kunnen zijn.
Om de instructie-codering compact te houden en de prestaties van de processor te verhogen, is het niet nodig dat instructies precies specificeren welk segment-register wordt gebruikt. Er wordt automatisch een default-register gekozen volgens de regels van tabel 7/4.1-6. Over het algemeen gebruiken data-verwijzingen de selector in het DS-register, stack-verwijzingen het SS-register en instructie-fetches het CS-register. De inhoud van de instructie-pointer levert de offset. Door speciale segment-override prefixes toe te passen is het nadrukkelijke gebruik van een bepaald segment-register mogelijk. Door de override prefixes kunnen ook de ES, FS en GS segment-registers worden gebruikt.

4.1 80386



Figuur 7/4.1-15: De adres-vertalingen.

Type of Memory Reference	Implied (Default) Segment Use	Segment Override Prefixes Possible
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA Instructions	SS	None
Source of POP, POPA, POPF, IRET, RET Instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS Instructions (DI is Base Register)	ES	None
Other Data References with Effective Address Using Base Register of:		
[EAX]	DS	CS, SS, ES, FS, GS
[EBX]	DS	CS, SS, ES, FS, GS
[ECX]	DS	CS, SS, ES, FS, GS
[EDX]	DS	CS, SS, ES, FS, GS
[ESI]	DS	CS, SS, ES, FS, GS
[EDI]	DS	CS, SS, ES, FS, GS
[EBP]	SS	CS, SS, ES, FS, GS
[ESP]	SS	CS, SS, ES, FS, GS

Tabel 7/4.1-6: Segment-register selectieregels.

Er zijn geen beperkingen ten aanzien van het overlappen van de basisadressen van segmenten. Van alle zes segmenten zou het basisadres dus nul kunnen zijn, bij een lineaire adresruimte van 4 GB. Hierdoor ontstaat een systeem waarin de virtuele asdresruimte gelijk is aan de lineaire adresruimte.

I/O-ruimte

De 386DX heeft twee aparte fysieke adresruimten: geheugen en I/O. Meestal worden periferieën in de I/O-ruimte geplaatst, hoewel de 386DX ook "memory-mapped" periferieën ondersteunt. De I/O-ruimte is 64 kB groot en kan worden verdeeld in 64 k 8 bit

poorten, 32 k 16 bit poorten, 16 k 32 bit poorten of elke andere combinatie die maximaal 64 kB groot is.
De 64 kB I/O-adresruimte slaat meer op fysiek geheugen dan op lineaire adressen, aangezien I/O-instructies niet door de segmentatie of paging-hardware heen gaan. De M/$\overline{IO}$-pen werkt als extra adreslijn, zodat de systeemontwerper gemakkelijk kan bepalen welk adres de processor benadert. De I/O-poorten zijn bereikbaar via de IN en OUT I/O-instructies, waarbij het poortadres als een 8 bit immediate constante in de instructie of in het DX-register staat.
Van alle 8 en 16 bit poortadressen zijn de hogere adreslijnen nul. Door I/O-instructies wordt de M/$\overline{IO}$-pen LAAG gemaakt.
De I/O-poortadressen van 00F8H tot en met 00FFH zijn door de fabrikant gereserveerd.

Interrupts

Interrupts en uitzonderingen

Interrupts en uitzonderingen veranderen het normale verloop van een programma om op externe gebeurtenissen te reageren of om fouten of buitengewone condities te melden.
Het verschil tussen interrupts en uitzonderingen is dat interrupts worden gebruikt om asynchrone externe gebeurtenissen te behandelen, terwijl uitzonderingen instructiefouten behandelen. Hoewel een programma een software-interrupt via een INT N instructie kan genereren, behandelt de processor software-interrupts als uitzonderingen.
Hardware-interrupts treden op als gevolg van een externe gebeurtenis en worden in twee typen verdeeld: maskeerbare en niet-maskeerbare. Interrupts worden na afloop van de lopende instructie afgehandeld. Nadat de interrupt-handler klaar is met de behandeling van de interrupt, gaat het programma verder met de instructie die onmiddellijk na de interrupt-instructie komt.
Uitzonderingen worden geklassificeerd als:

- faults;
- traps;
- aborts;

afhankelijk van de wijze waarop zij worden gerapporteerd en of herstarten van de instructie die de uitzondering veroorzaakte al dan niet wordt ondersteund.

- Faults
  Faults zijn uitzonderingen die worden gedetecteerd en behandeld voordat de veroorzakende instructie wordt uitgevoerd. Een fault treedt bijvoorbeeld op in een virtueel geheugensysteem wanneer de processor naar een niet bestaande pagina of segment verwijst. Het bedrijfssysteem zou dan de pagina of het segment van de vaste schijf ophalen, waarna de 386DX opnieuw met de instructie begint.
- Traps
  Traps zijn uitzonderingen die onmiddellijk na de uitvoering van de instructie die het probleem veroorzaakte worden uitgevoerd. Voorbeelden hiervan zijn interrupts die door de gebruiker worden gedefinieerd.
- Aborts
  Aborts zijn uitzonderingen die niet toestaan dat de instructie die het probleem veroorzaakte precies wordt gelokaliseerd. Aborts worden gebruikt om ernstige fouten te rapporteren, zoals een hardwarefout of ongeldige waarden in systeem-tabellen.

Samenvattend: wanneer een interrupt service-routine klaar is, gaat de uitvoering verder vanaf de instructie die onmiddellijk na de interrumperende instructie komt. Daarentegen wijst het return-adres van een uitzonderings fout-routine altijd naar de instructie die de uitzondering veroorzaakte, inclusief eventuele prefixes (zie ook tabel 7/4.1-7).

Interrupt-vectoren

De 386DX processor kan maximaal 256 verschillende interrupts en uitzonderingen afhandelen. Om de interrupts te kunnen bedienen moet een tabel met maximaal 256 vectoren worden gedefinieerd. De interrupt-vectoren zijn gewoon pointers naar de juiste interrupt service-routine.

4.1 80386

Function	Interrupt Number	Instruction Which Can Cause Exception	Return Address Points to Faulting Instruction	Type
Divide Error	0	DIV, IDIV	Yes	FAULT
Debug Exception	1	Any instruction	Yes	TRAP*
NMI Interrupt	2	INT 2 or NMI	No	NMI
One Byte Interrupt	3	INT	No	TRAP
Interrupt on Overflow	4	INTO	No	TRAP
Array Bounds Check	5	BOUND	Yes	FAULT
Invalid Op-Code	6	Any illegal instruction	Yes	FAULT
Device Not Available	7	ESC, WAIT	Yes	FAULT
Double Fault	8	Any instruction that can generate an Exception		ABORT
Coprocessor Segment Overrun	9	ESC	No	ABORT
Invalid TSS	10	JMP, CALL, IRET, INT	Yes	FAULT
Segment Not Present	11	Segment register instructions	Yes	FAULT
Stack Fault	12	Stack references	Yes	FAULT
General Protection Fault	13	Any memory reference	Yes	FAULT
Page Fault	14	Any memory access or code fetch	Yes	FAULT
Reserved for Future Use	15			
Coprocessor Error	16	ESC, WAIT	Yes	FAULT
Reserved for Future Use	17–31			
Two Byte Interrupt	0–255	INT n	No	TRAP

*Some debug exceptions may report both traps on the previous instruction and faults on the next instruction.

Tabel 7/4.1-7: Interrupt Vector-toekenningen.

In de Real Mode zijn de vectoren 4 byte grootheden: een code-segment plus een 16 bit offset. In de Protected Mode zijn interrupt-vectoren 8 byte grootheden die in een Interrupt Descriptor Table worden gezet. Van de 256 mogelijke interrupts zijn er 32 gereserveerd door de fabrikant.

Interrupt processing

Wanneer een interrupt optreedt, worden de volgende akties uitgevoerd:

- Eerst worden het lopende programma-adres en de vlaggen op de stack opgeborgen om hervatting van het onderbroken programma mogelijk te maken.
- Vervolgens wordt de processor van een 8 bit vector voorzien die de juiste ingang tot de interrupt-tabel identificeert. De tabel bevat het startadres van de interrupt service-routine.
- Daarna wordt de door de gebruiker verschafte interrupt service-routine uitgevoerd.
- Tenslotte wordt, na uitvoering van een IRET instructie, de oude toestand van de processor hersteld en gaat het programma verder bij de juiste instructie.

De 8 bit interrupt-vector wordt op verschillende manieren aan de processor verstrekt: uitzonderingen leveren de interrupt-vector intern; software INT instructies bevatten de vector of sluiten die in; maskeerbare hardware interrupts leveren de 8 bit vector via de interrupt-acknowledge bus-volgorde. Niet-maskeerbare hardware interrupts krijgen interrupt-vector 2 toegewezen.

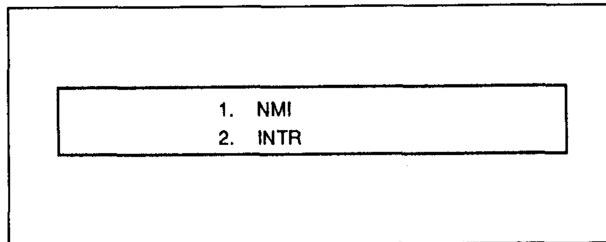
Maskeerbare interrupt
Als reactie op asynchrone externe hardware gebeurtenissen wordt meestal gereageerd met maskeerbare interrupts. Een hardware interrupt treedt op wanneer de INTR-lijn HOOG wordt, terwijl de Interrupt Flag-bit (IF) is vrijgegeven. De processor reageert alleen op interrupts tussen instructies, waarbij REPeat String-instructies een "interrupt window" hebben om het onderbreken van lange string-moves mogelijk te maken. Wanneer een interrupt optreedt, leest de processor een 8 bit vector die door de hardware wordt geleverd, om de interrumperende bron (één van de 224) te identificeren.
Het IF-bit in het EFLAG-register wordt gereset tijdens de bediening van een interrupt. Hierdoor wordt het bedienen van andere interrupts gedurende een interrupt service-routine onmogelijk. IF kan echter expliciet door de interrupt-handler worden gezet om het nesten van interrupts mogelijk te maken. Na de uitvoering van een IRET instructie wordt de originele toestand van het IF-bit teruggezet.

Niet-maskeerbare interrupt
Niet-maskeerbare interrupts vormen een manier om interrupts met zeer hoge prioriteit te behandelen. Het aktiveren van een power failure routine is een voorbeeld van zo'n NMI. Wanneer de NMI-ingang HOOG wordt getrokken veroorzaakt dit een interrupt met een intern geleverde vector-waarde van 2. In tegenstelling tot een normale hardware interrupt wordt voor een NMI geen interrupt bevestigings-volgorde uitgevoerd.
Tijdens de NMI service-procedure zal de 386DX geen andere NMI-verzoeken in behandeling nemen, totdat een IRET-instructie is uitgevoerd of totdat de processor is gereset. Als een NMI optreedt terwijl al een NMI wordt bediend, wordt de aanwezigheid ervan bewaard om te worden bediend na uitvoering van de eerste IRET-instructie. Het IF-bit wordt gecleared aan het begin van een NMI-interrupt om verdere INTR-interrupts tegen te houden.

Software interrupts
De software interrupt is het derde type interrupt/uitzondering waarover de 386DX processor kan beschikken. Een INT n instructie laat de processor de interrupt service-routine uitvoeren die door de n-de vector in de interrupt table wordt aangewezen. Een speciaal geval van de twee byte software interrupt INT n is de één byte INT 3 (breakpoint-)interrupt. Door deze één byte instructie in een programma op te nemen kan de gebruiker breakpoints aanbrengen om fouten op te sporen.

Prioriteiten van interrupts en uitzonderingen
Interrupts zijn extern opgewekte gebeurtenissen. Zowel maskeerbare- als niet-maskeerbare interrupts worden herkend op instructie-grenzen. Wanneer NMI en INTR beide op dezelfde instructiegrens worden herkend, voert de 386DX eerst de NMI service-routine uit. Als er daarna nog maskeerbare interrupts zijn, gaat de processor daar mee verder. Uitzonderingen zijn intern opgewekte gebeurtenissen. Ze worden door de 386DX gedetecteerd als tijdens de uitvoering van een instructie een problematische conditie wordt opgemerkt. De 386DX gaat dan onmiddellijk over op de bijbehorende uitzonderings service-routine. De processor is zo ingericht dat de instructie die de uitzondering veroorzaakte opnieuw kan worden gestart. Als de uitzonderings service-routine de problematische conditie heeft verholpen, zal deze instructie worden uitgevoerd zonder dezelfde uitzondering te veroorzaken.
Het is mogelijk dat één instructie meerdere uitzonderingen genereert (het overbrengen van een enkele operand kan bijvoorbeeld twee paginafouten genereren als de operand-lokatie twee "not present" pagina's omvat). Bij elke poging tot uitvoeren van de instructie wordt echter slechts één uitzondering gegenereerd. Iedere uitzonderings service-routine dient zijn bijbehorende uitzondering te corrigeren en de instructie opnieuw te beginnen.

4.1 80386



1. NMI
2. INTR

Tabel 7/4.1-8: Volgorde van behandeling van service-routines in het geval van gelijktijdige externe interrupts.

Op deze wijze worden uitzonderingen bediend totdat de instructie correct wordt uitgevoerd. Bij de uitvoering van instructies volgt de 386DX een vaste cyclus voor het checken van uitzonderingen (zie tabel 7/4.1-9). Deze cyclus wordt bij de uitvoering van elke instructie herhaald en dit gebeurt parallel aan de instructie-decodering en de uitvoering.

Opnieuw starten van de instructies

Bij de 386DX microprocessor worden alle instructies na het optreden van fouten opnieuw gestart. Als in de uit te voeren instructie een uitzondering wordt gedetecteerd (categorie 4 tot en met 10 in tabel 7/4.1-9) begint de 386DX met de bijbehorende uitzonderings service-routine. Alleen voor de in tabel 7/4.1-10 genoemde gevallen kan de 386DX niet opnieuw starten. Deze zijn echter eenvoudig te vermijden door een goed ontwerp van het bedrijfssysteem.

Dubbele fouten

Een dubbele fout (uitzondering 8) ontstaat wanneer de processor probeert een uitzonderings service-routine voor de segmentuitzonderingen (10, 11, 12 of 13) uit te voeren, maar daarbij een andere uitzondering dan een paginafout detecteert (uitzondering 14). Ook wordt een dubbele fout gegenereerd als de processor probeert de service-routine van een paginafout uit te voeren en daarbij een andere uitzondering detecteert. Bij een dubbele fout zal de 386DX de uitzonderings service-routine voor uitzondering 8 uitvoeren.

Consider the case of the Am386DXL microprocessor having just completed an instruction. It then performs the following checks before reaching the point where the next instruction is completed:

1. Check for Exception 1 Traps from the instruction just completed (single-step via Trap Flag or Data Breakpoints set in the Debug Registers).
2. Check for Exception 1 Faults in the next instruction (Instruction Execution Breakpoint set in the Debug Registers for the next instruction).
3. Check for external NMI and INTR.
4. Check for Segmentation Faults that prevented fetching the entire next instruction (Exceptions 11 and 13).
5. Check for Paging Faults that prevented fetching the entire next instruction (Exception 14).
6. Check for Faults decoding the next instruction (Exception 6 if illegal op-code; Exception 6 if in Real Mode or in Virtual 8086 Mode and attempting to execute an instruction for Protected Mode only (see Section Protection and I/O Permission Bitmap); or Exception 13 if instruction is longer than 15 bytes, or privilege violation in Protected Mode (i.e., not at IOPL or at CPL = 0)).
7. If WAIT op-code, check if TS = 1 and MP = 1 (Exception 7 if both are 1).
8. If ESCAPE op-code for numeric coprocessor, check if EM = 1 or TS = 1 (Exception 7 if either are 1).
9. If WAIT op-code or ESCAPE op-code for numeric coprocessor, check $\overline{ERROR}$ input signal (Exception 16 if $\overline{ERROR}$ input is asserted).
10. Check in the following order for each memory reference required by the instruction.
    a. Check for Segmentation Faults that prevent transferring the entire memory quantity (Exceptions 11, 12, 13).
    b. Check for Page Faults that prevent transferring the entire memory quantity (Exception 14).

Note that the order stated supports the concept of the paging mechanism being underneath the segmentation mechanism. Therefore, for any given code or data reference in memory, segmentation exceptions are generated before paging exceptions are generated.

Tabel 7/4.1-9: Volgorde van het checken van de uitzonderingen.

1. An instruction causes a task switch to a task whose Task State Segment (TSS) is partially not present. (An entire not present TSS is restartable.) Partially present TSS's can be avoided either by keeping the TSS's of such tasks present in memory or by aligning TSS segments to reside entirely within a single 4K page (for TSS segments of 4 Kb or less).
2. A coprocessor operand wraps around the top of a 64-Kb segment or a 4-Gb segment and spans three pages; and the page holding the middle portion of the operand is not present. This condition can be avoided by starting at a page boundary any segments containing coprocessor operands if the segments are approximately 64–200 Kb or larger (i.e., large enough for wraparound of the coprocessor operand to possibly occur).

Note that these conditions are avoided by using the operating system designs mentioned in this table.

Tabel 7/4.1-10: Condities die het herstarten van instructies voorkomen.

Flag Word	UUUU0002H	Note 1
Machine Status Word (CR0)	UUUUUUU0H	Note 2
Instruction Pointer	0000FFF0H	
Code Segment	F000H	Note 3
Data Segment	0000H	
Stack Segment	0000H	
Extra Segment (ES)	0000H	
Extra Segment (FS)	0000H	
Extra Segment (GS)	0000H	
DX Register	Component and Stepping ID	Note 5
All Other Registers	Undefined	Note 4

Notes:

1. EFLAGS Register. The upper 14 bits of the EFLAGS register are undefined, VM (Bit 17) and RF (Bit 16) and 0 as are all other defined flag bits.
2. CR0: (Machine Status Word). All of the defined fields in the CR0 are 0 (PG Bit 31, TS Bit 3, EM Bit 2, MP Bit 1, and PE Bit 0).
3. The code Segment Register (CS) will have its Base Address set to FFFF0000H and Limit set to 0FFFFH.
4. All undefined bits are Reserved for Future Use and should not be used.
5. DX register always holds component and stepping identifier (see Section Component and Revision Identifiers). EAX register holds self-test signature if self-test was requested (see Section Self-Test Signature).

Tabel 7/4.1-11: Register-waarden na het resetten.

Reset, initialisatie, testen en debuggen

Reset en initialisatie

Wanneer de processor is geïnitialiseerd of gereset hebben de registers de waarden zoals gegeven in tabel 7/4.1-11. De 386DX zal daarna beginnen met het uitvoeren van de instructies bij de top van het fysieke geheugen (op lokatie FFFFFFF0H). Wanneer de eerste InterSegment Jump of Call is uitgevoerd, gaan de adreslijnen A20 tot en met A31 LAAG bij geheugencycli die aan het CS zijn gerelateerd. De 386DX voert dan alleen instructies in de laagste één MB van het fysieke geheugen uit. Hierdoor kan de ontwerper een ROM gebruiken bovenop het fysieke geheugen om het systeem te initialiseren en Resets te behandelen.

Door RESET wordt de 386DX gedwongen om alle handelingen en de lokale busaktiviteiten te stoppen. Zolang RESET aktief is, vindt geen uitvoering of bus-aktiviteit plaats. Nadat RESET inactief is geworden duurt het 350 tot 450 CLK2-perioden voordat de processor weer begint met het uitvoeren van instructies.

Zelftest

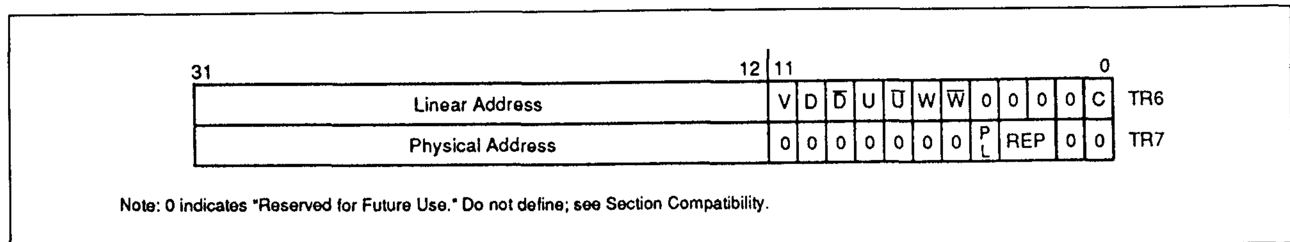
De 386DX heeft de mogelijkheid om zichzelf te testen. Hierbij wordt de functie van de gehele besturings-ROM en het grootste deel van de niet-bedrade logika getest. Met de zelftest kan ongeveer de helft van de 386DX microprocessor worden getest.

De zelftest begint als het signaal op de RESET-pen van HOOG naar LAAG gaat, terwijl $\overline{BUSY}$ LAAG is. De zelftest duurt ongeveer 2^{19} klokperioden (circa 26 ms voor een 20 MHz processor). Hierna wordt een reset uitgevoerd en begint de processor met de normale werking. De zelftest is geslaagd als de inhoud van het EAX-register nul is.

Testen van het TLB

De 386DX heeft een mechanisme voor het testen van het Translation Lookaside Buffer (TLB).

4.1 80386



Figuur 7/4.1-16: Test Registers in de 386DX.

Dit mechanisme is uniek voor de 386DX microprocessor, zodat het niet zeker is dat het op dezelfde manier ook in latere processoren wordt gebruikt.
Om de hardware voor het testen van het TLB vrij te geven en interferentie met de testdata die naar het TLB wordt geschreven te vermijden, moet de paginering worden uitgezet (PG = 0 in CR0).
Er zijn twee manieren om het TLB te testen: schrijven in het TLB en TLB-opzoekhandelingen uitvoeren. Voor het testen zijn twee Test Registers (figuur 7/4.1-16) beschikbaar. TR6 is het "test-commando register" en TR7 het "test-data register". Hieronder worden de velden in deze registers beschreven.

- C is het commando-bit. Om een immediate write in het TLB te veroorzaken moet een 0 naar dit bit in TR6 worden geschreven. Voor een immediate TLB lookup moet dit bit 1 worden.
- Linear Address is het identificatieveld (tag field) van het TLB. Bij het schrijven naar TLB wordt een TLB-ingangspunt op dit lineaire adres gezet, terwijl de rest van dat ingangspunt door de waarde in TR7 en de zojuist geschreven waarde in TR6 wordt bepaald. Bij het opzoeken in TLB wordt het TLB met deze waarde opgevraagd en als slechts één TLB ingangspunt hiermee overeenkomt, wordt de rest van de velden van TR6 en TR7 vanuit het passende TLB ingangspunt ingevuld.
- Physical Address is het dataveld van het TLB. Bij schrijven naar het TLB wordt het TLB ingangspunt dat in het lineaire adres van TR6 staat op deze waarde gezet. Bij het opzoeken in TLB wordt het dataveld (fysieke adres) uit het TLB hierin uitgelezen.
- PL: bij schrijven naar het TLB maakt PL = 1 dat het REP-veld van TR7 selecteert welk van de vier bijbehorende blokken van het TLB moet worden beschreven. Als PL = 0 is, mag de interne pointer in de paging unit echter selecteren welk TLB-blok wordt geschreven.
  Bij een TLB-lookup geeft het PL-bit aan of het opzoeken raak was (PL wordt 1) of mis (PL wordt 0).
- V is het Valid-bit voor deze TLB-toegang. De valid-bits kunnen ook worden gecleared door in CR3 te schrijven.
- D en D# zijn de dirty-bits voor/van de TLB-toegang.
- U en U# zijn de user-bits voor/van de TLB-toegang.
- W en W# zijn de writable bits voor/van de TLB-toegang.

Voor D, U en W wordt zowel het attribuut als het complement daarvan als identificatiebits meegeleverd om de optie "don't care" voor TLB-opzoekhandelingen mogelijk te maken. De betekenis van deze bitparen is te zien in tabel 7/4.1-12.

Ondersteuning bij het Debuggen
De 386DX processor heeft enkele voorzieningen die het foutzoeken vergemakkelijken. De drie categorieën on-chip debugginghulpen zijn:

- de code-executie breakpoint opcode 0CCH;
- de single-step mogelijkheid (met de TF-bit in het vlagregister);

– de code en data breakpoint mogelijkheid (met de Debug Registers DR0 tot en met DR3, DR6 en DR7.

X	X#	Effect During TLB Lookup	Value of Bit X after TLB Write
0	0	Miss All	Bit X becomes undefined
0	1	Match if X = 0	Bit X becomes 0
1	0	Match if X = 1	Bit X becomes 1
1	1	Match All	Bit X becomes undefined

Tabel 7/4.1-12: Betekenis van de D, U en W bitparen.

Breakpoint instructie

Voor gebruik in software debuggers is een één byte opcode breakpoint instructie beschikbaar. De breakpoint opcode is 0CCH die bij uitvoering een uitzondering 3 sprong genereert. In een foutzoek-programma kan de breakpoint instructie op alle gewenste code-executie breekpunten worden geplaatst.
De één byte breakpoint opcode is een alias voor de twee byte algemene software interrupt-instructie INT n (waarbij n = 3). Het enige verschil tussen INT 3 (0CCH) en INT n is, dat INT 3 nooit IOPL-gevoelig is (INT n is IOPL-gevoelig in de Protected Mode en in de Virtual 8086 Mode).

Single-Step Trap

Als bij het einde van een instructie wordt ontdekt dat de enkele-stap vlag (TF, bit 8) in het EFLAG register is gezet, treedt een single-step uitzondering op.
Deze single-step uitzondering krijgt vanzelf de uitzondering 1 vector. In de praktijk wordt meestal het TF-bit van een vlagregister-image op de stack van de debugger gezet, waarna het gebruikersprogramma de besturing overneemt en de vlag-image samen met de IRET-instructie wordt geladen. De enkele-stap sprong treedt op na de uitvoering van één instructie van het gebruikersprogramma.

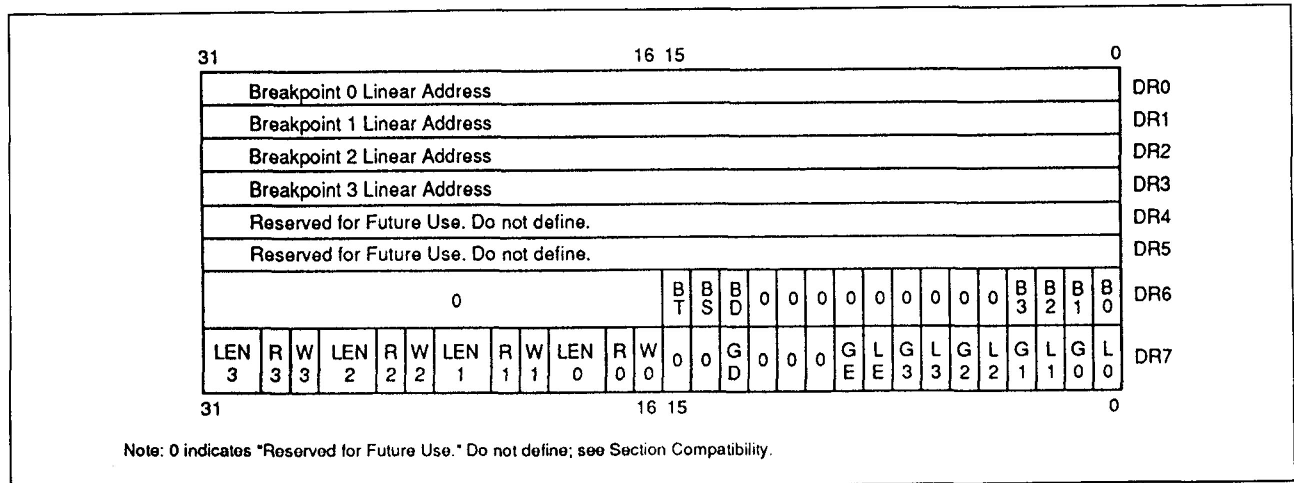
Debug Registers

De debug-registers maken het foutzoeken bij de 386DX zeer gemakkelijk, omdat hierdoor data-access breakpoints en code executie breakpoints mogelijk zijn. Aangezien de breakpoints door on-chip registers worden aangewezen, kan een instructie-breakpoint zowel in ROM-code worden geplaatst als in code die door verschillende taken wordt gedeeld (geen van beide wordt door de INT 3 breakpoint opcode ondersteund).
De 386DX microprocessor bevat zes Debug Registers, waarmee maximaal vier verschillende breakpoint-adressen, breakpoint-besturingsopties en lees-breakpoint-status kunnen worden gespecificeerd. Na het resetten bevinden de breekpunten zich oorspronkelijk in gesperde toestand. Daarom zullen geen breekpunten optreden zolang de debug-registers niet zijn geprogrammeerd. Breakpoints die in de debug-registers zijn ingesteld krijgen vanzelf een vector naar uitzondering 1.

Linear Address Breakpoint Registers (DR0 tot en met DR3)

Zoals in figuur 7/4.1-17 te zien is, kunnen door schrijven in de debug-registers DR0 tot en met DR3, maximaal vier breakpoint-adressen worden gespecificeerd.
Dit zijn 32 bit lineaire adressen.
De hardware van de 386DX vergelijkt voortdurend de lineaire breakpoint-adressen in DR0 tot en met DR3 met de lineaire adressen die worden opgewekt door het uitvoeren van de software (een lineair adres is de som van het effectieve adres en het 32 bit segment-basisadres). Merk op dat het lineaire adres gelijk is aan het fysieke adres als paginering niet is vrijgegeven. Is paginering wel vrijgegeven dan wordt het lineaire adres door de paging unit vertaald naar een 32 bit fysiek adres.

4.1 80386



Figuur 7/4.1-17: De debug-registers van de 386DX.

LENI Encoding	Breakpoint Field Width	Usage of Least Significant Bits in Breakpoint Address Register I, (I = 0–3)
00	1 byte	All 32-bits used to specify a single-byte breakpoint field.
01	2 bytes	A31–A1 used to specify a two-byte, word-aligned breakpoint field. A0 in Breakpoint Address Register is not used.
10	Undefined—do not use this encoding	
11	4 bytes	A31–A2 used to specify a four-byte, Dword-aligned breakpoint field. A0 and A1 in Breakpoint Address Register are not used.

Tabel 7/4.1-13: Codering van het LENi-veld.

Debug Control Register (DR7)

Door het Debug Control Register (DR7 in figuur 7/4.1-17) zijn verschillende debug-besturingsfuncties mogelijk, zoals het vrijgeven van de breakpoints en het instellen van andere besturings-opties voor de breakpoints. De velden in het debug-besturingsregister DR7 zijn:

- LENi;
- RWi;
- GD;
- GE;
- LE.

LENi (breakpoint lengte-specificatiebits)

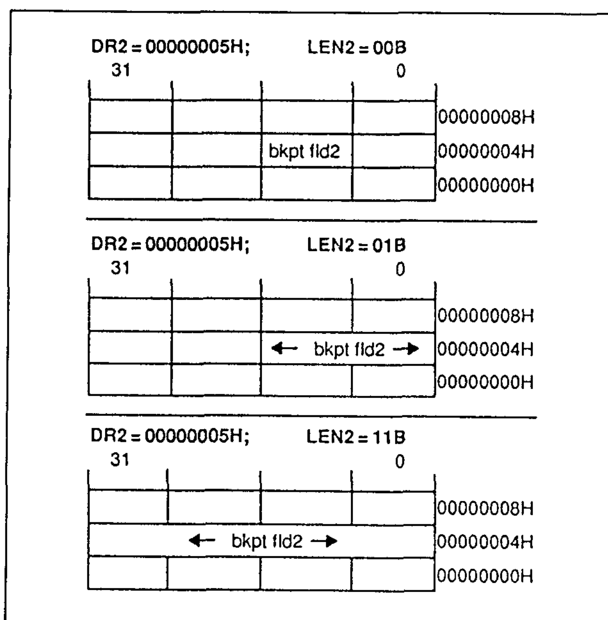
Voor elk van de vier breakpoints bestaat een 2 bit LEN-veld. LEN specificeert de lengte van het bijbehorende breakpoint-veld. De keuzes voor data-breakpoints zijn: 1 byte, 2 bytes of 4 bytes. Instructie-executie breakpoints moeten een lengte hebben van 1 (LENi = 00). Het coderen van het LENi-veld is te zien in tabel 7/4.1-13.

Het LENi-veld bepaalt de afmeting van het breakpoint-veld i door te regelen dat alle lage lineaire-adresbits in het breakpoint-adresregister worden gebruikt om het optreden van een breakpoint te detecteren. Daardoor zijn alle breakpoint-velden uitgelijnd: 2 byte breakpoint-velden beginnen op Word-grenzen en 4 byte breakpoint-velden beginnen op DWord-grenzen. In figuur 7/4.1-18 zijn enkele voorbeelden van verschillende breakpoint-velden (1, 2 en 4 bytes) te zien. Neem aan dat het breakpoint lineaire adres in DR2 00000005H is.

RWi (memory access qualifier bits)

Voor elk van de vier breakpoints bestaat een 2 bit RW-veld. Dit RW-veld specificeert wat moet gebeuren om het bijbehorende breakpoint te aktiveren (tabel 7/4.1-14). RW wordt

00 gecodeerd om een instructie-breakpoint in te stellen, terwijl 01 en 11 respectievelijk write-only en read/write-data breakpoints veroorzaken. Merk op dat instructie-executie breakpoints als faults worden behandeld (dat wil zeggen: vóór de instructie wordt uitgevoerd), maar dat data breakpoints als traps worden beschouwd (nadat de data-overdracht heeft plaats gevonden).



Figuur 7/4.1-18: Enkele verschillende breakpoint-velden (1, 2 en 4 bytes) op adres 00000005H.

RW Encoding	Usage Causing Breakpoint
00	Instruction execution only
01	Data writes only
10	Undefined—do not use this encoding
11	Data reads and writes only

Tabel 7/4.1-14: Aktivering van breakpoints door RW-codering.

Het gebruik van LENi en RWi om een Data-Breakpoint i in te stellen

Een data-breakpoint kan worden ingesteld door het lineaire adres in DR i te schrijven (i = 1 tot en met 3). Voor data-breakpoints kan RWi = 01 (write-only) of 11 (write/read) zijn. LEN kan = 00, 01 of 11 bedragen. Als een data-toegang geheel of gedeeltelijk binnen het data breakpoint-veld valt, de data breakpoint-conditie is opgetreden en het breakpoint vrijgegeven was, zal een uitzondering 1-trap optreden.

Het gebruik van LENi en RWi om een Instruction-Execution Breakpoint i in te stellen

Een instructie-executie breakpoint kan worden ingesteld door het adres van het begin van de instructie (inclusief eventuele prefixes) in DRi te schrijven (i = 1 tot en met 3). RWi en LEN moeten dan = 00 zijn. Als de instructie die op het breakpoint-adres begint op het punt staat te worden uitgevoerd, de instructie-executie breakpoint-conditie is opgetreden en het breakpoint is vrijgegeven, zal een uitzondering 1-fault optreden voordat de instructie wordt uitgevoerd.
Merk op dat een instructie-executie breakpoint-adres gelijk moet zijn aan het byte-adres waar de instructie begint (inclusief prefixes).

GD (Global Debug Register access detect)

Toegang tot de Debug Registers kan alleen worden verkregen in de Real Mode of op privilege-niveau 0 in de Protected Mode. Als het GD-bit is gezet levert dit extra beveiliging op tegen het benaderen van debug registers (zelfs in de Real Mode of op privilege-niveau 0 in de Protected Mode). Deze extra beveiliging maakt dat een software debugger (of ICE-386) volledige controle over het debug register kan hebben, als dat nodig is. Als het GD-bit is gezet, veroorzaakt een instructie die probeert een debug register te lezen of te beschrijven een uitzondering 1-fault. Wanneer de uitzondering 1-handler in werking komt, wordt het GD-bit automatisch gecleared.

4.1 80386

GE en LE (Globale en Lokale Exact data breakpoint match)
Als GE of LE is gezet zal elke data breakpoint-trap, na uitvoering van de instructie die de operand-overdracht veroorzaakte, exact worden gerapporteerd. Dit is mogelijk door de 386DX executie-unit te laten wachten tot de operand-overdracht geheel klaar is voordat de uitvoering van de volgende instructie begint. Als geen exacte breakpoint-match is geselecteerd, kan het zijn dat databreakpoints pas enkele instructies later of in het geheel niet worden gerapporteerd. Het wordt daarom aanbevolen de exact databreakpoint match vrij te geven wanneer een databreakpoint wordt ingesteld. Wanneer de 386DX processor een taakomschakeling uitvoert, wordt het LE-bit gecleared om te voorkomen dat in de nieuwe taak de exacte data breakpoint match is ingeschakeld.
Het GE-bit wordt bij taakomschakelingen niet beïnvloed. Het GE-bit ondersteunt exacte data breakpoint match die bij alle uitvoerende taken in het systeem ingeschakeld moet blijven. Merk op dat instruction execution breakpoints altijd exact worden gerapporteerd (of de exact data breakpoint match nu wel of niet is geselecteerd).

Gi en Li (Globale en Lokale breakpoint enable)
Als Gi of Li is gezet, is het bijbehorende breakpoint (zoals gedefinieerd door het lineaire adres in DRi, de lengte in LENi en de gebruikscriteria in RWi) vrijgegeven. Als de 386DX processor dan de i-de breakpointconditie detecteert, wordt de uitzondering 1-handler ingeschakeld. Als de 386DX een taakomschakeling naar een nieuw Task State Segment (TSS) uitvoert, worden alle Libits gecleared om valse uitzonderingen in de nieuwe taak te vermijden. Alle Gi-bits blijven bij taakomschakelingen onveranderd.

Debug Status Register (DR6)
Met behulp van het Debug Status-register (DR6 in figuur 7/4.1-17) kan de uitzondering 1-handler gemakkelijk bepalen waarom hij in werking was gekomen. Dit kan gebeuren als gevolg van één van de volgende gevallen:
- DR0 Breakpoint fault/trap;
- DR1 Breakpoint fault/trap;
- DR2 Breakpoint fault/trap;
- DR3 Breakpoint fault/trap;
- Single-step (TF) trap;
- Task switch trap;
- Fault als gevolg van een poging toegang te krijgen tot het debug-register bij GD = 1.

Het Debug Status-register bevat 1 bit vlaggen voor elk van de mogelijke gebeurtenissen die op uitzondering 1 betrekking hebben. Merk op dat sommige hiervan faults zijn en andere traps. De vlaggen in DR6 worden gezet door de hardware, maar nooit gecleared door de software. Uitzondering 1-handler software dient DR6 te clearen voordat wordt teruggekeerd naar het gebruikers-programma om te voorkomen dat later verwarring ontstaat bij het identificeren van de oorzaak van uitzondering 1. De velden binnen het Debug Status Register zijn:
- Bi (debug fault/trap als gevolg van breakpoint 0 tot en met 3)
  Er zijn vier breakpoint indicatie-vlaggen (B0 tot en met B3) die één voor één overeenkomen met de breakpoint-registers in DR0 tot en met DR3. Wanneer de conditie optreedt zoals die bij DRi, LENi en RWi is beschreven, wordt een vlag Bi gezet. Als Gi of Li is gezet en als het i-de breakpoint wordt gedetecteerd, zal de processor de uitzondering 1-handler oproepen. De uitzondering wordt als een fault behandeld bij een instructie-executie breakpoint, of als een trap bij een data breakpoint. Telkens wanneer de hardware een matchconditie op een vrijgegeven breakpoint i detecteert, wordt een vlag-bit Bi gezet.
- BD (debug fault als gevolg van een poging toegang te krijgen tot registers als GD is gezet)
  Dit bit wordt gezet als de uitzondering 1-handler in werking werd gesteld bij een poging om in de debug-registers te lezen of te schrijven wanneer GD was gezet.

Wanneer dit gebeurt, wordt het GD-bit automatisch gecleared bij inschakeling van de uitzondering 1-handler.

- BS (debug trap als gevolg van single-step)
  Dit bit wordt gezet als de uitzondering 1-handler werd ingeschakeld omdat het TF-bit in het flag-register was gezet (voor single-stepping).
- BT (debug trap als gevolg van taak-omschakeling)
  Dit bit wordt gezet als de uitzondering 1-handler werd ingeschakeld als gevolg van een taak-omschakeling. Ten opzichte van de taak-omschakeling wordt de operatie beschouwd als trap.

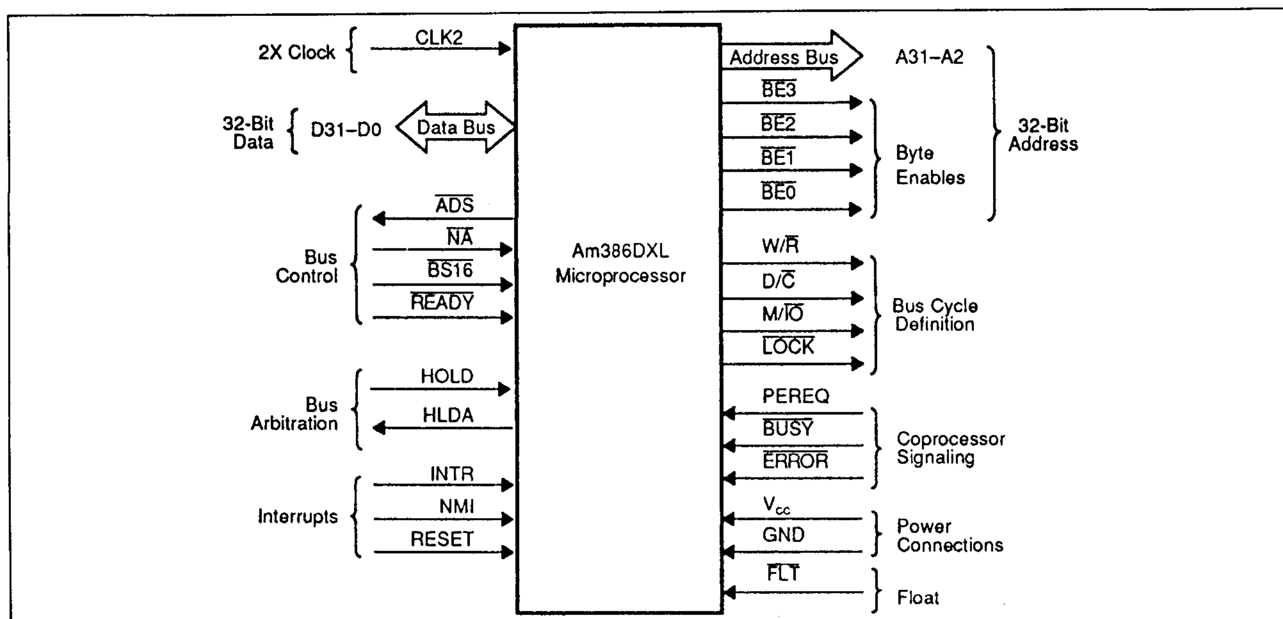
Gebruik van de
Resume Flag (RF) in het Flag Register
De Resume Flag (RF) in het flag-word kan een instructie-executie breakpoint onderdrukken, wanneer de uitzondering 1-handler terugkeert naar een gebruikers-programma op een gebruikers-adres dat tevens een instructie-executie breakpoint is.

Functionele gegevens

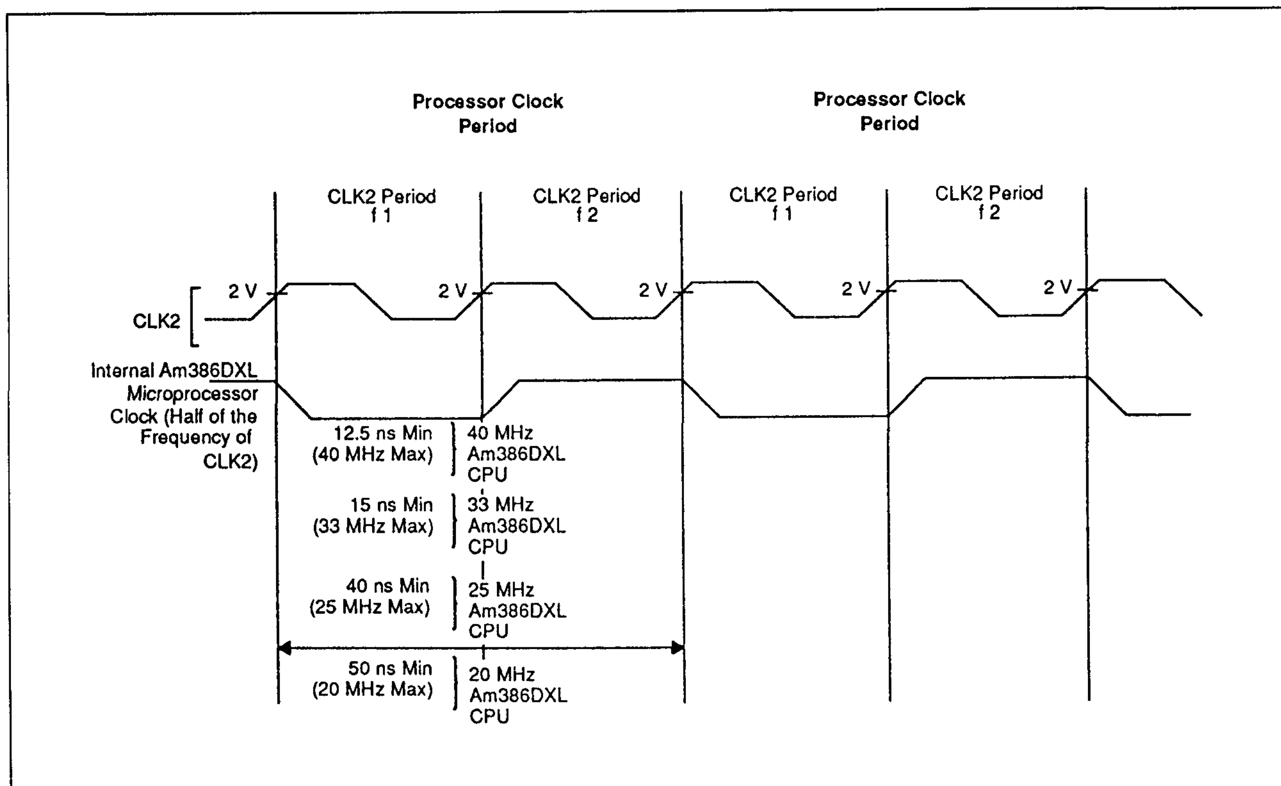
Inleiding
De 386DX microprocessor kan op eenvoudige wijze op de externe hardware worden aangesloten. Er zijn aparte parallelle bussen voor data en adressen. De databus is 32 bit breed en bidirectioneel. De adresbus levert 32 bit adreswaarden in de meest direct bruikbare vorm voor de high-speed lokale bus: 4 individuele byte enable-signalen en 30 hogere bits als binaire waarde.
De data- en adresbussen worden geïnterpreteerd en bestuurd met hun bijbehorende besturingssignalen. Door "dynamic databus sizing" kan de processor overweg met gemengde 32 en 16 bit externe bussen. Als een 16 bit bus-afmeting is geselecteerd, doet de 386DX automatisch alle benodigde aanpassingen, inclusief een eventuele extra 16 bit buscyclus om de overdracht af te maken. Randapparatuur van 8 bit kan op 32 bit of 16 bit bussen worden aangesloten zonder verlies van prestatievermogen. Bovendien is een nieuwe adrespijplijn optie aangebracht voor een aanzienlijk verbeterd geheugengebruik (tenminste één wait state minder dan anders).
De gepijplijnde bus is ook zeer geschikt voor om-en-om geheugenontwerpen (interleaved). Wanneer adres-pijplijnen door de externe hardware wordt gevraagd, zal de 386DX de adres- en buscyclus-definitie van de volgende buscyclus geven (als die intern beschikbaar is), zelfs terwijl op het bevestigen van de lopende cyclus wordt gewacht. Nietgepijplijnde adres-timing is echter ideaal voor externe cache-ontwerpen, aangezien het cache-geheugen meestal snel genoeg zal zijn. Om het ontwerpen zo flexibel mogelijk te maken, kan de pijplijn-optie op een cyclus-voor-cyclus basis worden geselecteerd. Het mechanisme van informatieoverdracht, van processor naar systeem of omgekeerd, berust op de buscyclus van de processor. De buscycli van de 386DX processor voeren data-transporten uit in minimaal twee clockperioden. De 32 bit databus heeft bij bijvoorbeeld 33 MHz dus een overdracht bandbreedte van 66 MB/s. Als de externe hardware een cyclus echter niet bevestigt, zal deze worden verlengd. Op het juiste tijdstip wordt de bevestiging gesignaleerd door een signaal op de $\overline{READY}$-ingang van de 386DX processor. De 386DX kan de besturing van de lokale bussen overgeven aan andere schakelingen, zoals DMA-kanalen. Wanneer de besturing is afgestaan, is HLDA de enige uitgang die nog door de 386DX wordt aangedreven, zodat de processor bijna geheel geïsoleerd wordt van zijn systeem.

Beschrijving van de signalen
Hieronder volgt een korte, groepsgewijze beschrijving van de in- en uitgangssignalen van de 386DX (zie figuur 7/4.1-19). Een streepje boven de naam van het signaal geeft aan dat de aktieve of opgelegde toestand optreedt als het signaal LAAG is.

4.1 80386



Figuur 7/4.1-19: Functioneel gegroepeerde signalen.



Figuur 7/4.1-20: CLK2 signaal en interne processorklok.

Ontbreekt dit streepje, dan is het signaal aktief bij een HOOG niveau.Soms hebben de signaal-beschrijvingen betrekking op AC-timing parameters, zoals "t_{25} Reset Setup Time".

Clock (CLK2)
CLK2 levert de fundamentele timing voor de processor. Deze wordt intern door twee gedeeld om de inwendige processorklok op te wekken die voor het uitvoeren van de instructies dient.
De interne klok heeft twee fasen: "fase 1" en "fase 2".
Elke CLK2-periode is een fase van de interne klok (zie figuur 7/4.1-20). Als dat nodig is kan de fase van de interne processorklok worden gesynchroniseerd op een bekende fase door ervoor te zorgen dat de dalende flank van het RESET-signaal binnen de setup- en hold-tijden t_{25} en t_{26} valt.

Databus (D0 tot en met D31)
Deze 3-state bidirectionele signalen vormen het algemeen bruikbare datapad tussen de 386DX processor en andere schakelingen. Databus ingangen en uitgangen zijn logisch "1" als ze HOOG zijn. De databus kan data overbrengen via 32 bit of 16 bit bussen door gebruik te maken van de $\overline{BS16}$-ingang. Voor een correcte uitlezing van data moet worden voldaan aan de data setup- en hold-tijden t_{21} en t_{22}. Bovendien moeten alle databuspennen aan het einde van elke leescyclus op een geldig logisch niveau staan (HOOG of LAAG), als $\overline{READY}$ aanwezig is. Gedurende elke schrijf-operatie (en ook tijdens halt-cycli en shutdown-cycli) drijft de 386DX altijd alle 32 signalen van de databus aan, zelfs als de bus op dat moment 16 bit breed is.

Adresbus ($\overline{BE0}$ tot en met $\overline{BE3}$ en A2 tot en met A31)
Deze 3-state uitgangen leveren fysieke geheugenadressen of I/O-adressen. Met de adresbus kan 4 GB fysieke geheugenruimte (00000000H tot en met FFFFFFFFH) en 64 kB I/O-adresruimte (00000000H tot en met 0000FFFFH) voor geprogrammeerde I/O worden geadresseerd. Automatisch gegenereerde I/O-overdrachten voor communicatie tussen 386DX-processor en coprocessor maken gebruik van de I/O-adressen 800000F8H tot en met 800000FFH, zodat het coprocessor-selectsignaal eenvoudig met A31 = HOOG plus M/$\overline{IO}$ = LAAG kan worden opgewekt.
De Byte Enable-uitgangen $\overline{BE0}$ tot en met $\overline{BE3}$ geven direct aan welke bytes van de 32 bit databus betrokken zijn bij de lopende overdracht. Dit is zeer handig voor externe hardware.

- $\overline{BE0}$ heeft betrekking op D0 t/m D7
- $\overline{BE1}$ heeft betrekking op D8 t/m D15
- $\overline{BE2}$ heeft betrekking op D16 t/m D23
- $\overline{BE3}$ heeft betrekking op D24 t/m D31

Het aantal ingeschakelde Byte Enables geeft de fysieke grootte aan van de operand die wordt overgebracht (1, 2, 3 of 4 bytes).
Wanneer een geheugen-schrijfcyclus of een I/O-schrijfcyclus aan de gang is, bezet de over te brengen operand alleen maar de hoogste 16 bits van de databus (D16 t/m D31), terwijl gedupliceerde data gelijktijdig op de overeenkomstige laagste 16 bits van de databus staat (D0 t/m D15). Deze duplicatie wordt uitgevoerd voor optimale schrijfprestaties op 16 bit bussen. Het patroon van de schrijf-data duplicatie is een functie van de ingeschakelde Byte Enables tijdens schrijfcycli (zie tabel 7/4.1-15).

Buscyclus-definitiesignalen (W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$ en $\overline{LOCK}$)
Deze 3-state uitgangen definiëren de soort buscyclus die wordt uitgevoerd. W/$\overline{R}$ maakt onderscheid tussen lees- en schrijfcycli, D/$\overline{C}$ doet dat voor data- en besturings (control) cycli, M/$\overline{IO}$ bepaalt of het geheugen- of I/O-cycli zijn en $\overline{LOCK}$ onderscheidt geblokkeerde (locked) en niet-geblokkeerde buscycli. De eerste drie worden de primaire buscyclus-definitiesignalen genoemd, omdat ze geldig zijn zodra de $\overline{ADS}$ (Address Status output) is ingeschakeld.

4.1 80386

Am386DXL CPU Byte Enables				Am386DXL CPU Write Data				Automatic Duplication?
$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$	D31–D24	D23–D16	D15–D8	D7–D0	
High	High	High	Low	Undef	Undef	Undef	A	No
High	High	Low	High	Undef	Undef	B	Undef	No
High	Low	High	High	Undef	C	Undef	C	Yes
Low	High	High	High	D	Undef	D	Undef	Yes
High	High	Low	Low	Undef	Undef	B	A	No
High	Low	Low	High	Undef	C	B	Undef	No
Low	Low	High	High	D	C	D	C	Yes
High	Low	Low	Low	Undef	C	B	A	No
Low	Low	Low	High	D	C	B	Undef	No
Low	Low	Low	Low	D	C	B	A	No

Key: D = Logical Write Data D31–D24 B = Logical Write Data D15–D8
C = Logical Write Data D23–D16 A = Logical Write Data D7–D0

Tabel 7/4.1-15: Schrijf-data duplicatie als functie van $\overline{BE0}$ tot en met $\overline{BE3}$.

M/$\overline{IO}$	D/$\overline{C}$	W/$\overline{R}$	Bus Cycle Type		Locked?
Low	Low	Low	Interrupt Acknowledge		Yes
Low	Low	High	Does Not Occur		—
Low	High	Low	I/O Data Read		No
Low	High	High	I/O Data Write		No
High	Low	Low	Memory Code Read		No
High	Low	High	Halt: Address = 2 $\overline{BE0}$ High $\overline{BE1}$ High $\overline{BE2}$ Low $\overline{BE3}$ High A31–A2 Low	Shutdown: Address = 0 $\overline{BE0}$ Low $\overline{BE1}$ High $\overline{BE2}$ High $\overline{BE3}$ High A31–A2 Low	No
High	High	Low	Memory Data Read		Some Cycles
High	High	High	Memory Data Write		Some Cycles

Tabel 7/4.1-16: Definitie van buscycli.

LOCK wordt geldig op het moment dat de eerste geblokkeerde buscyclus begint (als er sprake is van pijplijnen, kan dit later zijn dan het moment dat $\overline{ADS}$ wordt ingeschakeld).
In tabel 7/4.1-16 worden de exacte buscyclus-definities getoond. Merk op dat één combinatie van (W/$\overline{R}$, D/$\overline{C}$ en M/$\overline{IO}$ nooit wordt gegeven bij ingeschakelde $\overline{ADS}$ (treedt nooit op: "does not occur").

Bus-besturingssignalen ($\overline{ADS}$, $\overline{READY}$, $\overline{NA}$ en $\overline{BS16}$)
De volgende signalen maken dat de processor aangeeft wanneer een buscylus is begonnen en stellen andere systeem-hardware in staat om adres-pijplijning, databusbreedte en beëindiging van een buscyclus te regelen.

Address Status ($\overline{ADS}$)
Deze 3-state uitgang geeft aan dat een geldige buscyclus-definitie en een geldig adres (W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{BE0}$ tot en met $\overline{BE3}$ en A2 tot en met A31) op de pennen van de microprocessor staan. Het is ingeschakeld tijdens de T1 en T2P bus-toestanden.

Transfer Acknowledge ($\overline{READY}$)
Deze ingang geeft aan dat de lopende buscyclus klaar is en dat de aktieve bytes ($\overline{BE0}$

tot en met $\overline{BE3}$ en $\overline{BS16}$) zijn geaccepteerd of aangebracht. Wanneer $\overline{READY}$ bij monstering tijdens een leescyclus of een interrupt acknowledge-cyclus blijkt te zijn ingeschakeld, latcht de 386DX de ingangsdata en beëindigt de cyclus. $\overline{READY}$ wordt bij de eerste buscyclus van alle buscycli genegeerd en bij iedere daarna komende buscyclus bemonsterd totdat het is ingeschakeld. $\overline{READY}$ moet ten slotte worden ingeschakeld om elke buscyclus, inclusief Halt-indicatie en Shutdown-indicatie buscycli, te bevestigen. Wanneer $\overline{READY}$ wordt bemonsterd, moet altijd aan de setup en hold-tijden t_{19} en t_{20} worden voldaan.

Next Address Request ($\overline{NA}$)
Dit wordt gebruikt om adres-pijplijnen aan te vragen. Het signaal geeft aan dat het systeem klaar staat om nieuwe waarden van $\overline{BE0}$ tot en met $\overline{BE3}$, A2 tot en met A31, W/$\overline{R}$, D/$\overline{C}$ en M/$\overline{IO}$ van de processor te accepteren, zelfs als het einde van de lopende cyclus nog niet is bevestigd met $\overline{READY}$. Voor een goede werking moet $\overline{NA}$ altijd aan de setup en hold-tijden t_{15} en t_{16} voldoen.

Bus Size 16 ($\overline{BS16}$)
$\overline{BS16}$ maakt het mogelijk dat de 386DX direct op 32 bit en 16 bit databussen kan worden aangesloten. Door deze ingang in te schakelen gebruikt de lopende buscyclus alleen de laagste helft (D0 tot en met D15) van de databus, hetgeen overeenkomt met $\overline{BE0}$ en $\overline{BE1}$ = LAAG. Het inschakelen van $\overline{BS16}$ heeft geen extra effect als tijdens de lopende buscyclus alleen $\overline{BE0}$ en $\overline{BE1}$ zijn ingeschakeld. Als $\overline{BS16}$ echter tijdens buscycli met ingeschakelde $\overline{BE2}$ of $\overline{BE3}$ wordt ingeschakeld, doet de 386DX processor automatisch aanpassingen voor een juiste overdracht van de hogere byte(s) met behulp van slechts de fysieke data-signalen D0 tot en met D15.
Als de operand beide helften van de databus omspant en $\overline{BS16}$ is ingeschakeld, voert de 386DX automatisch nog een 16 bit buscyclus uit. $\overline{BS16}$ moet, om goed te kunnen werken, aan de setup en hold-tijden t_{17} en t_{18} voldoen.
386DX microprocessor I/O-cycli worden automatisch gegenereerd voor coprocessor-communicatie. Aangezien de 386DX 32 bit waarden moet overbrengen tussen zichzelf en de 387DX, mag $\overline{BS16}$ gedurende 387DX communicatie-cycli NIET zijn ingeschakeld.

Bus Arbitration Signals (HOLD, HLDA)
In dit gedeelte wordt beschreven hoe de processor afstand doet van de besturing van zijn lokale bussen, wanneer een andere busmaster daarom vraagt.

Bus Hold Request (HOLD)
Dit signaal geeft aan dat een andere schakeling dan de 386DX processor meester van de bus wil worden. HOLD moet ingeschakeld blijven zolang een andere schakeling lokale busmaster is. HOLD wordt niet erkend als RESET is ingeschakeld. Wordt RESET ingeschakeld bij reeds ingeschakelde HOLD, dan heeft RESET voorrang en zet de bus in een leegloop positie. HOLD is een niveau-gevoelige synchrone ingang en moet altijd aan de setup en hold-tijden t_{23} en t_{24} voldoen.

Bus Hold Acknowledge (HLDA)
Deze uitgang wordt ingeschakeld om aan te geven dat de 386DX microprocessor de besturing van zijn lokale bus heeft afgestaan als antwoord op het HOLD-signaal en in de bus Hold Acknowledge toestand verkeert. De Hold Acknowledge toestand biedt bijna volledige signaal-isolatie, waarbij HLDA het enige signaal is dat door de 386DX wordt aangedreven. De overige uitgangs- of bidirectionele signalen (D0 tot en met D31, $\overline{BE0}$ tot en met $\overline{BE3}$, A2 tot en met A31, W/$\overline{R}$, D/$\overline{C}$, M/$\overline{IO}$, $\overline{LOCK}$, en $\overline{ADS}$) zijn in de hoog-impedante toestand, zodat de aanvragende busmaster ze kan besturen. Bij sommige signalen kunnen optrekweerstanden wenselijk zijn om onbedoelde activiteit te voorkomen, wanneer ze niet worden aangedreven. Ook wordt één stijgende flank op de

NMI-ingang tijdens Hold Acknowledge onthouden om te worden behandeld na weghalen van het HOLD-signaal.
Behalve het normale gebruik van Hold Acknowledge met DMA-controllers of masterperiferieën, is de bijna volledige isolatie bijzonder aantrekkelijk voor het testen van het systeem.

Coprocessor Interface Signals (PEREQ, BUSY en ERROR)
Hierna volgt de beschrijving van signalen die bij de numerieke coprocessor behoren. Naast de databus, adresbus en buscyclusdefinitiesignalen regelen de volgende signalen de communicatie tussen de 386DX microprocessor en de 387DX processoruitbreiding.

Coprocessor Request (PEREQ)
Als dit ingangssignaal aanwezig is, betekent dit dat een coprocessor de 386DX vraagt om een data-operand van/naar geheugen te sturen. Als antwoord daarop verstuurt de 386DX informatie tussen de coprocessor en geheugen.
Omdat de 386DX de coprocessor-opcode die wordt uitgevoerd intern heeft opgeslagen, voert hij de gevraagde data-overdracht in de juiste richting naar het juiste geheugenadres uit. PEREQ is niveau-gevoelig en mag asynchroon van het CLK2-signaal zijn.

Coprocessor Busy (BUSY)
Dit signaal geeft aan dat de coprocessor nog steeds bezig is met het uitvoeren van een instructie, zodat hij nog geen nieuwe kan aannemen. Wanneer de 386DX een coprocessor-instructie tegenkomt die op de numerieke stack werkt (bijvoorbeeld load, pop of een rekenkundige operatie), of de WAIT-instructie, wordt deze ingang eerst automatisch bemonsterd totdat dit signaal wegvalt. Het bemonsteren van de BUSY-ingang voorkomt dat overrun optreedt op een eerdere coprocessor-instructie.
De FNINIT en FNCLEX coprocessor-instructies mogen wel worden uitgevoerd terwijl BUSY is ingeschakeld, aangezien deze instructies worden gebruikt voor de initialisatie van de coprocessor en het clearen van uitzonderingen.
BUSY is niveau-gevoelig en mag asynchroon ten opzichte van het CLK2-signaal zijn.
BUSY heeft nog een extra functie: als BUSY op de dalende flank van RESET LAAG is, voert de 386DX een interne zelftest uit.

Coprocessor Error (ERROR)
Dit ingangssignaal geeft aan dat de vorige coprocessor-instructie een type coprocessorfout genereerde die niet door het besturingsregister van de coprocessor werd gemaskeerd. Deze ingang wordt automatisch door de 386DX bemonsterd als die een coprocessor-instructie tegenkomt. Wordt dit signaal aangetroffen, dan genereert de 386DX microprocessor een uitzondering-16 om toegang te krijgen tot de software die de fout moet behandelen.
Verschillende coprocessor-instructies (over het algemeen die de numerieke fout-vlaggen in de coprocessor clearen of de toestand van de coprocessor opslaan) worden uitgevoerd zonder dat de 386DX een uitzondering-16 genereert, zelfs bij ingeschakelde ERROR. Deze instructies zijn FNINIT, FNCLEX, FSTSW, FSTSWAX, FSTCW, FSTENV, FSAVE, FESTENV en FESAVE.
ERROR is niveau-gevoelig en mag asynchroon van het CLK2-signaal worden aangeboden.

Interrupt Signalen (INTR, NMI, RESET)
In het volgende worden ingangssignalen behandeld die de uitvoering van de lopende instructiestroom van de processor kunnen onderbreken of ophouden.

Maskable Interrupt Request (INTR)
Als dit signaal aanwezig is, geeft het aan dat een interrupt-service wordt verlangd die met het IF-bit in het 386DX Flag Register kan worden gemaskeerd. Wanneer de 386DX op het INTR-signaal reageert, voert hij twee in-

terrupt-acknowledge buscycli uit, waarbij hij aan het einde van de tweede een 8 bit interrupt-vector op D0 tot en met D7 latcht om de interrumperende bron te identificeren.
INTR is niveau-gevoelig en mag asynchroon van het CLK2-signaal worden aangeboden. Om er zeker van te zijn dat een INTR-verzoek wordt herkend, moet het aanwezig blijven tot de eerste interruptacknowledge buscyclus begint.

Non-Maskable Interrupt Request (NMI)
Dit signaal geeft aan dat een interrupt-service wordt gevraagd die niet door software kan worden gemaskeerd. De niet-maskeerbare interrupt-request verloopt altijd volgens de pointer of gate in positie 2 van de interrupt-tabel. Door de vaste waarde hiervan worden bij het behandelen van NMI geen interrupt-acknowledge cycli uitgevoerd.
NMI is stijgende flank-gevoelig en mag asynchroon van het CLK2-signaal optreden. Om zeker herkend te worden moet NMI minstens 8 CLK2-perioden afwezig zijn en daarna gedurende minstens 8 CLK2-perioden aanwezig.
Zodra de behandeling van NMI is begonnen, worden geen verdere NMI's verwerkt tot na de volgende IRET-instructie die zich normaal aan het eind van de NMI service-routine bevindt. Als NMI echter vóór die tijd opnieuw verschijnt, zal één stijgende flank van NMI worden onthouden om na de volgende IRET-instructie te worden behandeld.

Reset (RESET)
Dit ingangssignaal stelt iedere lopende operatie uit en zet de 386DX processor in een bekende reset-toestand. De processor wordt gereset door RESET gedurende minimaal 15 CLK2-perioden in te schakelen (80 of meer CLK2-perioden voordat zelftest wordt gevraagd). Als RESET aanwezig is, worden alle overige ingangspennen genegeerd en worden de andere buspennen in een vrijloop-toestand gezet (zie tabel 7/4.1-17). Als RESET en HOLD beide op een bepaald tijdstip aanwezig zijn, gaat RESET voor, zelfs als de 386DX zich in een Hold Acknowledge-toestand bevond voordat RESET werd ingeschakeld.

Pin Name	Signal Level During Reset
$\overline{ADS}$	High
D31–D0	High Impedance
$\overline{BE3}$–$\overline{BE0}$	Low
A31–A2	High
W/$\overline{R}$	Low
D/$\overline{C}$	High
M/$\overline{IO}$	Low
$\overline{LOCK}$	High
HLDA	Low

Tabel 7/4.1-17: Toestand van de pennen (bij vrijlopende bus) gedurende Reset.

RESET is niveau-gevoelig en moet synchroon lopen met het CLK2-signaal. Indien gewenst kunnen de fase van de interne processorklok en de gehele toestand van de 386DX processor worden gesynchroniseerd op externe schakelingen door te zorgen dat de dalende flank van het reset-signaal aan de setup en holdtijden t_{25} en t_{26} voldoet.
Tenslotte zijn in tabel 7/4.1-18 de eigenschappen van alle 386DX-signalen te zien.

Bus-overdracht mechanisme
Alle data-overdrachten vinden plaats als resultaat van één of meer buscycli. Logische data-operands met byte, woord en dubbelwoord lengten mogen worden verplaatst zonder beperkingen met betrekking tot fysieke adres-uitlijning. Elke willekeurige byte-grens mag worden gebruikt, hoewel voor niet-uitgelijnde operand-overdrachten twee of zelfs drie fysieke buscycli nodig zijn.
De 386DX adressignalen zijn ontworpen om de externe systeem-hardware zo eenvoudig mogelijk te maken. Hogere orde adresbits worden bediend door A2 tot en met A31. Het lagere orde adres in de vorm van $\overline{BE0}$ tot en met $\overline{BE3}$ selecteert direct de vier bytes van de 32 bit databus. Informatie over de fysieke operand-grootte is daarom bij elke buscylus in de meest bruikbare vorm impliciet aanwezig.

4.1 80386

Signal Name	Function	Active State	Input/ Output	Input Synch or Asynch to CLK2	Output High Impedance During HLDA?
CLK2	Clock	—	I	—	—
D31–D0	Data Bus	High	I/O	S	Yes
$\overline{BE3}$–$\overline{BE0}$	Byte Enables	Low	O	—	Yes
A31–A2	Address Bus	High	O	—	Yes
W/$\overline{R}$	Write-Read Indication	High	O	—	Yes
D/$\overline{C}$	Data-Control Indication	High	O	—	Yes
M/$\overline{IO}$	Memory-I/O Indication	High	O	—	Yes
$\overline{LOCK}$	Bus Lock Indication	Low	O	—	Yes
$\overline{ADS}$	Address Status	Low	O	—	Yes
$\overline{NA}$	Next Address Request	Low	I	S	—
$\overline{BS16}$	Bus Size 16	Low	I	S	—
$\overline{READY}$	Transfer Acknowledge	Low	I	S	—
HOLD	Bus Hold Request	High	I	S	—
HLDA	Bus Hold Acknowledge	High	O	—	No
PEREQ	Coprocessor Request	High	I	A	—
$\overline{BUSY}$	Coprocessor Busy	Low	I	A	—
$\overline{ERROR}$	Coprocessor Error	Low	I	A	—
INTR	Maskable Interrupt Request	High	I	A	—
NMI	Non-Maskable Intrpt Request	High	I	A	—
RESET	Reset	High	I	S	—

Tabel 7/4.1-18: Samenvatting van de 386DX microprocessor-signalen.

Byte Enable Signal	Associated Data Bus Signals
$\overline{BE0}$	D7–D0 (Byte 0—least significant)
$\overline{BE1}$	D15–D8 (Byte 1)
$\overline{BE2}$	D23–D16 (Byte 2)
$\overline{BE3}$	D31–D24 (Byte 3—most significant)

Tabel 7/4.1-19: Byte Enables met bijbehorende Data- en Operand-bytes.

De Byte Enable-uitgangen $\overline{BE0}$ tot en met $\overline{BE3}$ worden ingeschakeld wanneer hun bijbehorende databus-bytes bij de lopende buscyclus betrokken zijn (zie tabel 7/4.1-19). Gedurende een buscyclus is elk patroon van onafgebroken ingeschakelde Byte Enable-uitgangen mogelijk, maar geen patronen waarbij een uitgeschakelde Byte Enable twee of meer ingeschakelde Enables scheidt. De adresbits A0 en A1 van het fysieke basisadres van de operand kunnen, indien nodig (bijvoorbeeld voor een MULTIBUS I of MULTIBUS II interface) worden gecreëerd als een functie van de laagste-orde ingeschakelde Byte Enable. Dit wordt duidelijk gemaakt aan de hand van tabel 7/4.1-20 en figuur 7/4.1-21. Elke buscyclus is samengesteld uit tenminste twee bustoestanden. Elke bustoestand heeft één processor-clockperiode nodig. Extra bustoestanden die aan een enkele buscyclus worden toegevoegd, worden wait states genoemd. Aangezien een buscyclus minimaal twee bustoestanden nodig heeft (gelijk aan twee processor-clockperioden) kan data tussen externe schakelingen en de 386DX processor met een maximale snelheid van één 4 byte Dword per twee processor-clockperioden worden overgebracht. Dit komt overeen met een maximale busbandbreedte van 66 MB/s als de 386DX met een 33 MHz processor-clock werkt.

Am386DXL CPU Address Signals								
A31	....	A2			$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$
	Physical Base Address							
A31	....	A2	A1	A0				
A31	....	A2	0	0	X	X	X	Low
A31	....	A2	0	1	X	X	Low	High
A31	....	A2	1	0	X	Low	High	High
A31	....	A2	1	1	Low	High	High	High

Tabel 7/4.1-20: Het genereren van A0 tot en met A31 met behulp van $\overline{BE0}$ tot en met $\overline{BE3}$ en A2 tot en met A31.

Geheugen en I/O-ruimten

Inleiding

Buscycli kunnen toegang hebben tot fysieke geheugenruimte of I/O-ruimte. Periferieschakelingen in het systeem kunnen memory-mapped, I/O-mapped of beide zijn.

Zoals in figuur 7/4.1-22 te zien is, lopen de fysieke adressen van 00000000H tot FFFFFFFFH (4 GB) en I/O adressen van 00000000H tot 0000FFFFH (64 kB) voor geprogrammeerde I/O. Merk op dat de I/O-adressen die worden gebruikt door de automatische I/O-cycli voor coprocessor-communicatie liggen tussen 800000F8H en 800000FFH (verder dan het adresbereik van geprogrammeerde I/O) om gemakkelijk een coprocessor chip-select signaal met behulp van de A31 en M/$\overline{IO}$-signalen op te wekken.

Organisatie van geheugen en I/O

Het datapad van de 386DX naar geheugen en I/O-ruimten kan 32 bit of 16 bit breed zijn. Wanneer het 32 bit breed is, zijn het geheugen en de I/O-ruimten natuurlijk als arrays van fysieke 32 bit dubbelwoorden (Dwords) georganiseerd. Elk geheugen- of I/O Dword heeft vier individueel adresseerbare bytes op aaneensluitende byte-adressen. De laagst geadresseerde byte is verbonden met de data-signalen D0 tot en met D7; de hoogst geadresseerde byte met D24 tot en met D31. De 386DX processor beschikt over een busbesturingsingang $\overline{BS16}$ waardoor ook directe verbinding met 16 bit geheugen of I/O-ruimten, georganiseerd als een reeks 16 bit woorden, mogelijk is. Cycli naar 32 bit en 16 bit geheugen of I/O-schakelingen mogen in elke willekeurige volgorde voorkomen, aangezien de $\overline{BS16}$-besturing gedurende iedere buscyclus wordt bemonsterd.

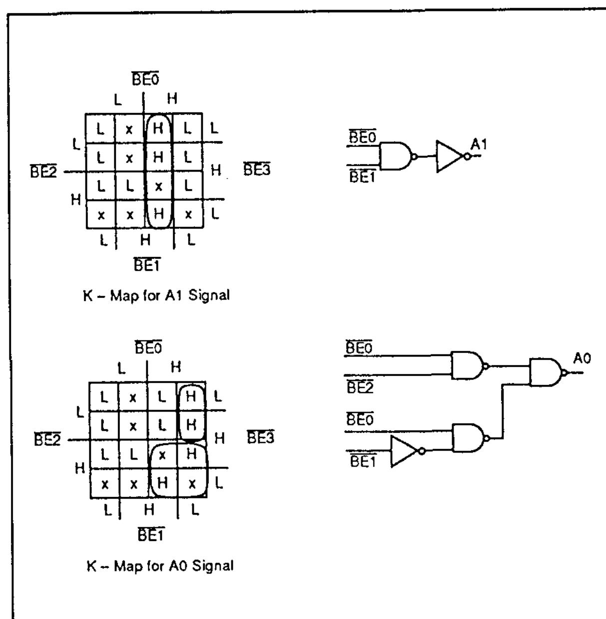
Dynamische Databus breedte-instelling

Het dynamisch instellen van de busbreedte (dynamic bus sizing) is een eigenschap die directe aansluiting op 32 bit of 16 bit databussen voor geheugen of I/O mogelijk maakt. Een enkele processor kan zodoende met bussen van beide grootten werken. De overdrachten van en naar 32 of 16 bit poorten worden ondersteund door gedurende iedere buscyclus dynamisch de busbreedte te bepalen. Bij elke buscyclus kan een adresdecodeerschakeling of de slaaf zelf $\overline{BS16}$ voor 16 bit poorten inschakelen of $\overline{BS16}$ weghalen voor 32 bit poorten.
Als $\overline{BS16}$ is ingeschakeld, zet de processor automatisch operand-transfers die groter zijn dan 16 bits of verkeerd uitgelijnde 16 bit

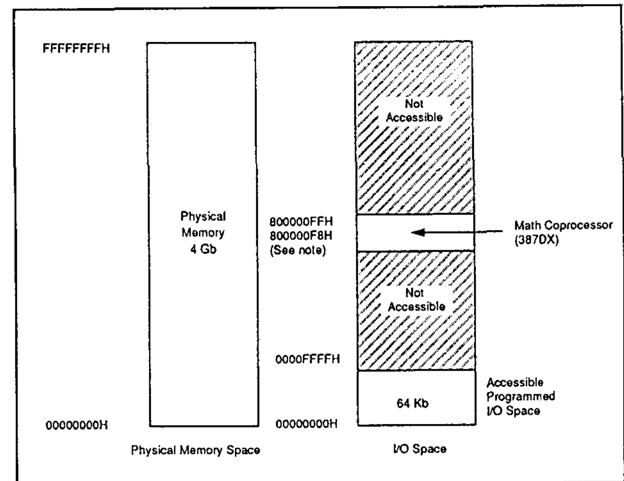
4.1 80386

transfers om in de vereiste twee of drie overdrachten. Als $\overline{BS16}$ aanwezig is, vonden alle operand-overdrachten fysiek plaats op D0 tot en met D15. Daarom worden 16 bit geheugens of I/O-schakelingen alleen aangesloten op de datalijnen D0 tot en met D15 en zijn er geen extra transceivers nodig.
Het inschakelen van $\overline{BS16}$ beïnvloedt de processor alleen als $\overline{BE2}$ en/of $\overline{BE3}$ tijdens de lopende cyclus zijn ingeschakeld. Als alleen D0 tot en met D15 bij de overdracht betrokken zijn, heeft het inschakelen van $\overline{BS16}$ geen invloed, aangezien de transfer normaal via een 16 bit bus kan plaats vinden, of $\overline{BS16}$ nu wel of niet aanwezig is.
Er zijn twee situaties waarbij de processor wel wordt beïnvloed door het inschakelen van $\overline{BS16}$, afhankelijk van welke Byte Enables tijdens de lopende buscyclus zijn ingeschakeld:

- Alleen hoogste helft:
  Alleen $\overline{BE2}$ en/of $\overline{BE3}$ ingeschakeld.
- Hoogste en laagste helft:
  Tenminste $\overline{BE1}$ en $\overline{BE2}$ ingeschakeld (en misschien ook $\overline{BE0}$ en $\overline{BE3}$).



Figuur 7/4.1-21: Benodigde logika om A0 en A1 te genereren uit $\overline{BE0}$ tot en met $\overline{BE3}$.



Figuur 7/4.1-22: Fysiek geheugen en I/O-ruimten.

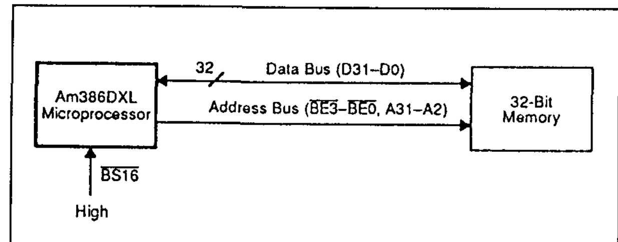
- Effect van $\overline{BS16}$ inschakelen gedurende "alleen hoogste helft" leescycli:
  Het inschakelen van $\overline{BS16}$ maakt dan dat de 386DX processor data leest op de laagste 16 bits van de databus en de hoogste 16 bits negeert. Data die van D16 tot en met D31 zou worden gelezen (zoals aangegeven met $\overline{BE2}$ en $\overline{BE3}$) wordt dan gelezen op D0 tot en met D15.
- Effect van $\overline{BS16}$ inschakelen gedurende "alleen hoogste helft" schrijfcycli:
  Het inschakelen van $\overline{BS16}$ beïnvloedt de 386DX processor in dit geval niet. Wanneer tijdens een schrijfcyclus alleen $\overline{BE2}$ en/of $\overline{BE3}$) worden ingeschakeld, dupliceert de 386DX processor de datasignalen D16 tot en met D31 altijd naar D0 tot en met D15. Daarom zijn geen verdere akties nodig om deze schrijf-operaties op 32 bit of 16 bit bussen uit te voeren.
- Effect van $\overline{BS16}$ inschakelen gedurende de " hoogste en laagste helft" leescycli:
  Het inschakelen van $\overline{BS16}$ maakt in dit geval dat de 386DX processor twee 16 bit leescycli uitvoert om de complete fysieke operand-transfer te volbrengen. De bytes 0 en 1 (zoals aangegeven met $\overline{BE0}$ en $\overline{BE1}$) worden met de eerste cyclus op D0 tot en met D15 gelezen en de bytes 2 en 3 met de tweede cyclus (ook weer op D0

tot en met D15). D16 tot en met D31 worden bij beide 16 bit cycli genegeerd, terwijl BE0 en BE1 gedurende de tweede 16 bit cyclus worden uitgeschakeld (zie ook figuur 7/4.1-32, cycli 2 en 2a).

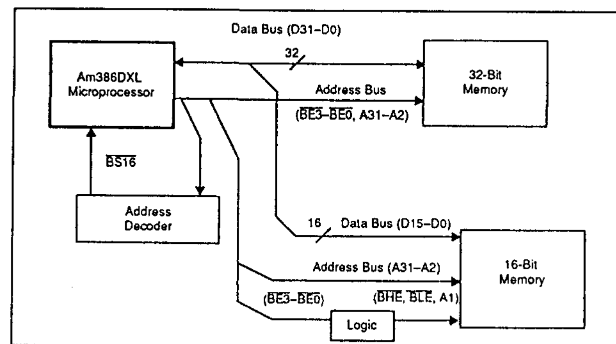
- Effec t van BS16 inschakelen gedurende "hoogste en laagste helft" schrijfcycli:
  Het inschakelen van BS16 laat nu voor een complete fysieke operand-transfer de 386DX processor twee 16 bit schrijfcycli uitvoeren.
  Alle bytes zijn bij de eerste schrijfcyclus beschikbaar om externe hardware in staat te stellen de bytes 0 en 1 (zoals aangegeven met BE0 en BE1) te ontvangen op D0 tot en met D15. Bij de tweede cyclus dupliceert de 386DX de bytes 2 en 3 (zoals aangegeven met BE2 en BE3) op D0 tot en met D15. BE0 en BE1 worden gedurende de tweede 16 bit cyclus altijd uitgeschakeld (zie ook figuur 7/4.1-32, cycli 1 en 1a).

Het aansluiten op 32 en 16 bit geheugens

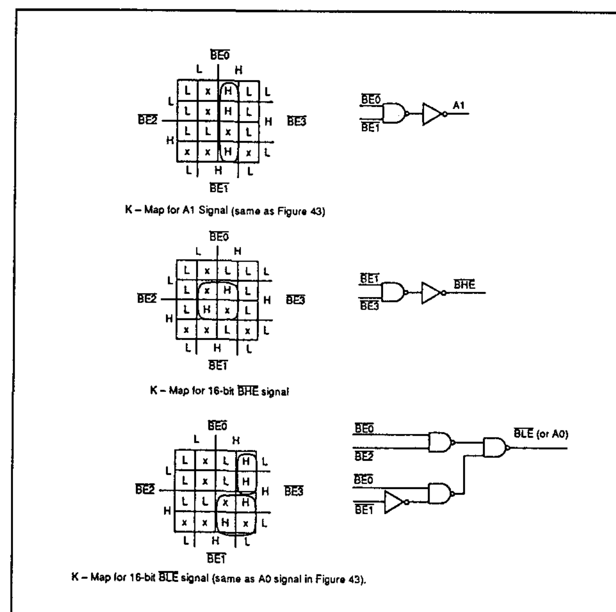
Bij 32 bit brede geheugens zoals in figuur 7/4.1-23 begint elk fysieke dubbelwoord (Dword) op een byte-adres dat een veelvoud van 4 is. De lijnen A2 tot en met A31 worden direct gebruikt als een Dword-select en BE0 tot en met BE3 als byte-selects. BS16 wordt bij alle 32 bit buscycli genegeerd.
Wanneer het systeem ook 16 bit brede arrays bevat, zoals in figuur 7/4.1-24, begint elk 16 bit fysieke woord op een adres dat een veelvoud van 2 is. Merk op dat het adres wordt gedecodeerd om BS16 alleen gedurende 16 bit buscycli in te schakelen. Als het gewenst is om gepijplijnde adressen met 16 bit geheugens te gebruiken, worden BE0 tot en met BE3 en W/R ook gedecodeerd om te bepalen wanneer BS16 moet zijn ingeschakeld.
De adreslijnen A2 tot en met A31 zijn direct bruikbaar om 32 bit en 16 bit schakelingen te adresseren. Voor het adresseren van 16 bit schakelingen zijn ook A1 en twee byte enable-signalen nodig.



Figuur 7/4.1-23: De 386DX processor, aangesloten op een 32 bit geheugen.



Figuur 7/4.1-24: De 386DX processor, aangesloten op 32 bit en 16 bit geheugen.



Figuur 7/4.1-25: Logika om A1, BHE en BLE te genereren voor 16 bit bussen.

4.1 80386

Am386DXL CPU Signals				16-Bit Bus Signals			Comments
$\overline{\text{BE3}}$	$\overline{\text{BE2}}$	$\overline{\text{BE1}}$	$\overline{\text{BE0}}$	A1	$\overline{\text{BHE}}$	$\overline{\text{BLE}}$ (A0)	
H*	H*	H*	H*	X	X	X	X—no active bytes
H	H	H	L	L	H	L	
H	H	L	H	L	L	H	
H	H	L	L	L	L	L	
H	L	H	H	H	H	L	
H*	L*	H*	L*	X	X	X	X—not contiguous bytes
H	L	L	H	L	L	H	
H	L	L	L	L	L	L	
L	H	H	H	H	L	H	
L*	H*	H*	L*	X	X	X	X—not contiguous bytes
L*	H*	L*	H*	X	X	X	X—not contiguous bytes
L*	H*	L*	L*	X	X	X	X—not contiguous bytes
L	L	H	H	H	L	L	
L*	L*	H*	L*	X	X	X	X—not contiguous bytes
L	L	L	H	L	L	H	
L	L	L	L	L	L	L	

$\overline{\text{BLE}}$ asserted when D7–D0 of 16-bit bus is active.
$\overline{\text{BHE}}$ asserted when D15–D8 of 16-bit bus is active.
A1 Low for all even words; A1 High for all odd words.

Key: X = Don't care
H = High voltage level
L = Low voltage level
* = A non-occurring pattern of Byte Enables; either none are asserted or the pattern has Byte Enables asserted for non-contiguous bytes.

Tabel 7/4.1-21: Het genereren van A1, $\overline{\text{BHE}}$ en $\overline{\text{BLE}}$ om 16 bit schakelingen te adresseren.

Om het A1 signaal en twee Byte Enables voor een 16 bit toegang te genereren, moeten $\overline{\text{BE0}}$ tot en met $\overline{\text{BE3}}$ worden gedecodeerd zoals in tabel 7/4.1-21 te zien is. Merk op dat sommige $\overline{\text{BE0}}$ tot en met $\overline{\text{BE3}}$-combinaties nooit door de 386DX processor worden gegenereerd (en tot "don't care" condities in de decoder leiden). In $\overline{\text{BE0}}$ tot en met $\overline{\text{BE3}}$-decoders kan van deze niet-optredende combinaties zo goed mogelijk gebruik worden gemaakt (zie figuur 7/4.1-25).

Operand uitlijning

Met de flexibele geheugen-adressering is het mogelijk om een logische operand te verstuven die groter is dan één fysiek Dword of woord.

Voorbeelden hiervan zijn 32 bit Dword-operands die beginnen op adressen die niet deelbaar zijn door 4, of een 16 bit woord-operand die is verdeeld over twee fysieke Dwords van het geheugen-array.

Operand-uitlijning en databus-afmeting bepalen wanneer meerdere buscycli nodig zijn. Tabel 7/4.1-22 beschrijft de transfer-cycli die voor alle combinaties van logische operand-lengten, uitlijningen en databus-afmetingen worden gegenereerd. Wanneer meerdere buscycli nodig zijn om een multi-byte logische operand over te dragen, worden de hoogste bytes het eerst verzonden, maar als $\overline{\text{BS16}}$ is ingeschakeld en twee 16 bit cycli worden uitgevoerd, worden de laagste bytes het eerst verstuurd.

	Byte-Length of Logical Operand								
	1	2				4			
Physical Byte Addres in Memory (low-order bits)	xx	00	01	10	11	00	01	10	11
Transfer Cycles over 32-Bit Data Bus	b	w	w	w	hb,* lb	d	hd l3	hw, lw	h3, lb
Transfer Cycles over 16-Bit Data Bus	b	w	lb, hb	w	hb, lb	lw, hw	hb, lb, mw	hw, lw	mw, hb, lb

Key: b = Byte transfer
w = Word transfer
l = low-order portion
m = mid-order portion
3 = 3-byte transfer
d = Dword transfer
h = high-order portion
x = Don't care
= $\overline{BS16}$ asserted causes second bus cycle.

*For this case, 8086, 8088, 80186, 80188, 80286 transfer lb first, then hb.

Tabel 7/4.1-22: Overdracht buscycli voor Bytes, Words en Dwords.

Beschrijving van bus-functies

Inleiding

De 386DX microprocessor heeft aparte parallelle bussen voor data en adressen. De databus is 32 bit breed en werkt in twee richtingen (bidirectioneel). De adresbus levert een 32 bit waarde door 30 signalen te gebruiken voor de 30 hoogste adresbits en 4 Byte Enable-signalen om direct de aktieve bytes aan te geven. Deze bussen worden geïnterpreteerd en gecontroleerd via verschillende bijbehorende definitie- of besturingssignalen. De definitie van elke buscyclus wordt gegeven door drie definitiesignalen: M/$\overline{IO}$, W/$\overline{R}$ en D/$\overline{C}$. Tegelijkertijd is een geldig adres aanwezig op de byte enable-lijnen $\overline{BE0}$ tot en met $\overline{BE3}$ en andere adres-lijnen A2 tot en met A31. Een statussignaal $\overline{ADS}$ geeft aan wanneer de 386DX een nieuwe buscyclus-definitie en adres laat verschijnen. De adresbus, databus en alle bijbehorende besturingssignalen worden tezamen "de bus" genoemd.

Buscycli

Als de bus aktief is, wordt één van de volgende buscycli uitgevoerd:
- uitlezen van geheugenruimte;
- geblokkeerd uitlezen van geheugenruimte;
- schrijven naar geheugenruimte;
- geblokkeerd schrijven naar geheugenruimte;
- lezen uit I/O-ruimte (of coprocessor);
- schrijven naar I/O-ruimte (of coprocessor);
- interrupt acknowledge;
- aangeven van halt of shutdown.

Tabel 7/4.1-16 laat zien hoe de buscyclus definitie-signalen voor elke buscyclus worden gecodeerd.

De databus heeft een dynamische breedteregeling voor 32 en 16 bit (16 bit busafmeting wordt aangegeven met $\overline{BS16}$). Wanneer de bus niet bezig is om één van de hiervoor vermelde aktiviteiten uit te voeren, bevindt hij zich óf in de vrijloop-toestand óf in de Hold Acknowledge toestand, hetgeen door externe schakelingen gedetecteerd kan worden.

Bus-toestand

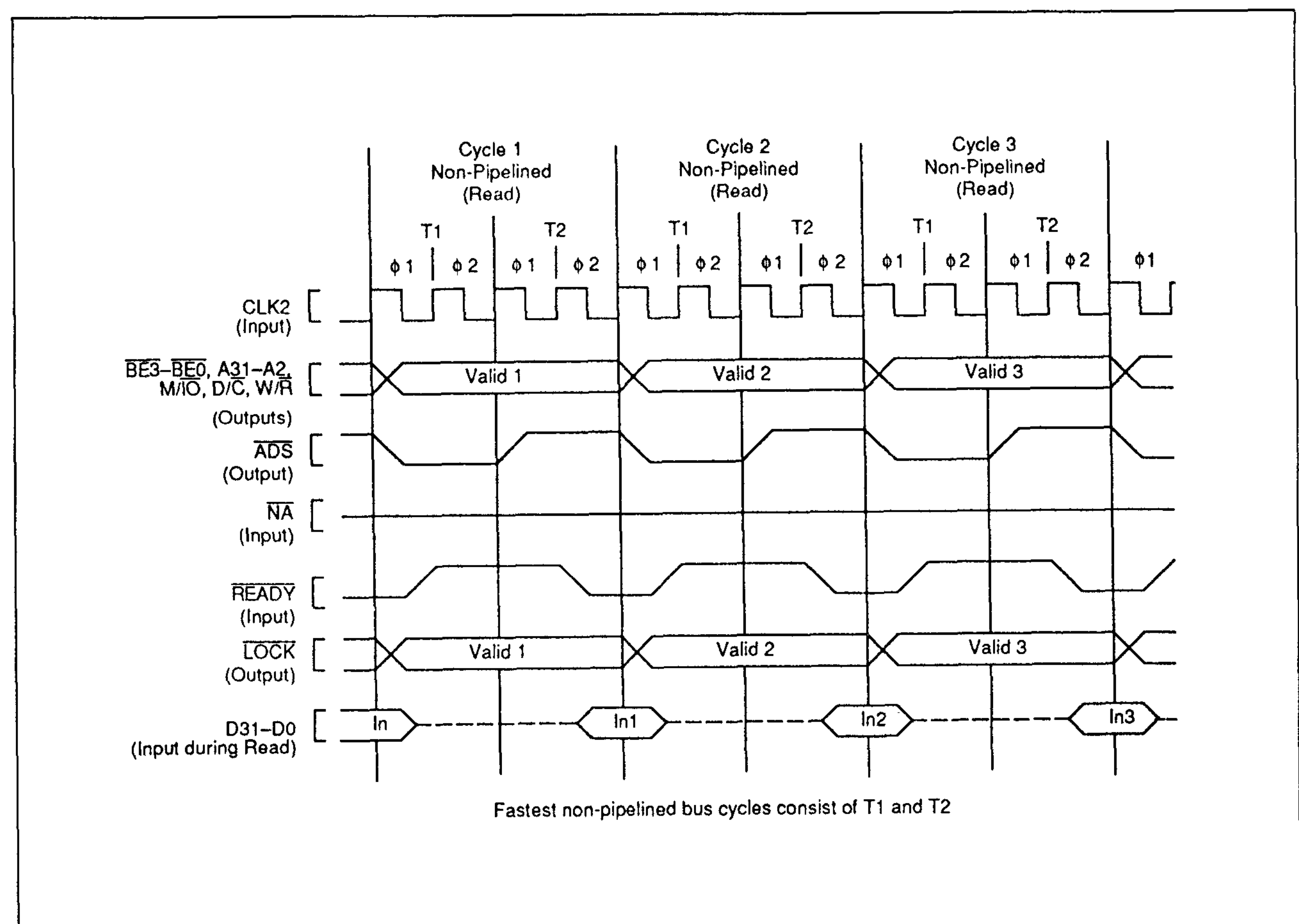
De kortste tijdseenheid van bus-aktiviteit is een bus-toestand. Een bus-toestand is één processor-clockperiode (twee CLK2 perioden) lang. Een complete data-overdracht geschiedt in een buscyclus die is samengesteld uit twee of meer bus-toestanden. De snelste 386DX buscyclus heeft slechts twee bus-toestanden nodig. In figuur 7/4.1-26 zijn bijvoorbeeld drie opeenvolgende bus-leescycli te zien, die elk uit twee bus-toestanden

(T1 en T2) bestaan. Elk geheugen- of I/O-adres kan met een dergelijke, uit twee toestanden bestaande, buscyclus worden bereikt als de externe hardware tenminste snel genoeg is. Iedere buscyclus gaat door totdat hij door de externe systeem-hardware wordt bevestigd (acknowledge) door middel van de READY-ingang. Het bevestigen van de buscyclus aan het einde van de eerste T2 heeft de kortste buscyclus tot gevolg. Als READY echter niet onmiddellijk verschijnt, worden T2-toestanden voor onbepaalde tijd herhaald totdat het READY-signaal bij bemonstering wel wordt aangetroffen.

Adres-pijplijning

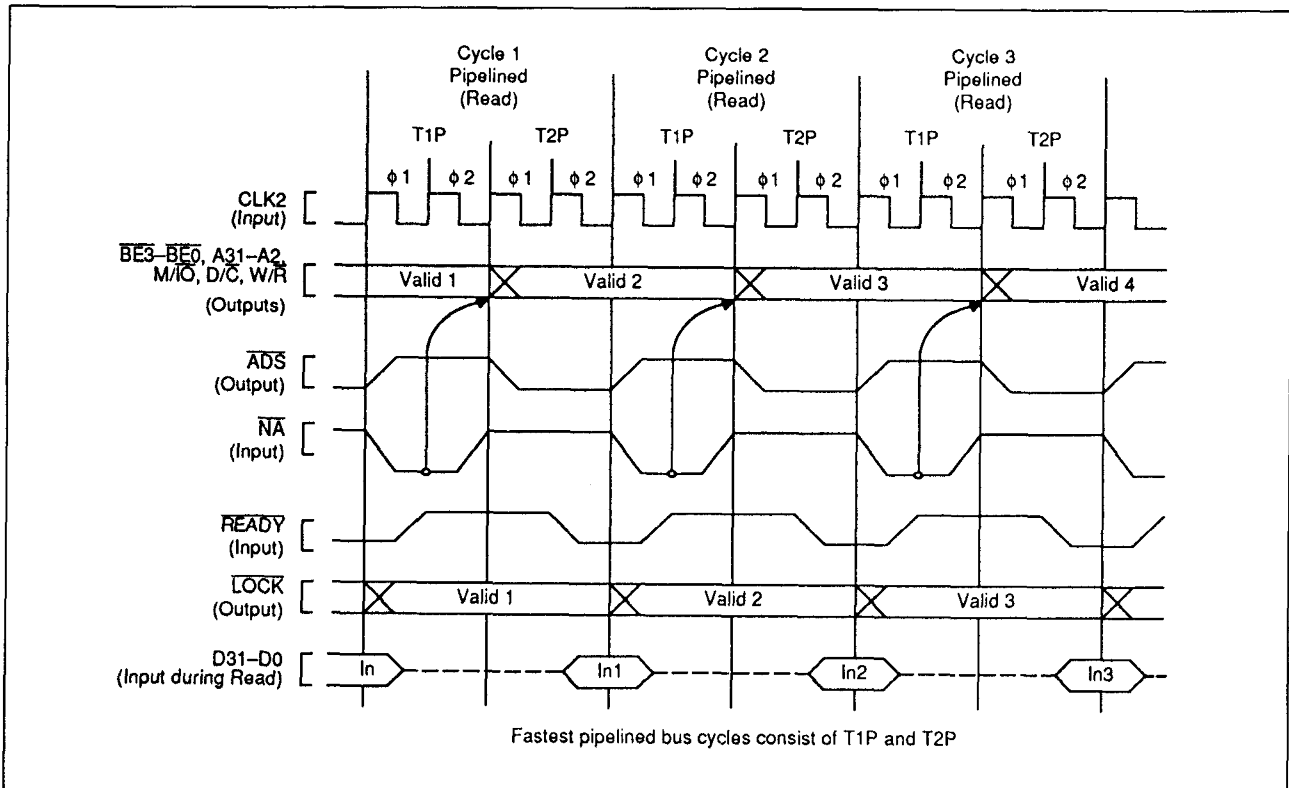
De adres-pijplijn optie maakt verschillende buscyclus-timingen mogelijk. Met behulp van de Next Address-ingang (NA) kan op een cyclus-voor-cyclus basis worden gekozen voor gepijplijnde of niet-gepijplijnde adrestiming. Wanneer niet voor adres-pijplijnen is gekozen, blijven de lopende adres- en buscyclus-definitie gedurende de gehele buscyclus stabiel.

Wanneer adres-pijplijning is geselecteerd, zijn het adres (BE0 tot en met BE3, A2 tot en met A31) en de definitie (W/R, D/C en M/IO) van de volgende cyclus beschikbaar vóór het einde van de lopende cyclus. Om hun beschikbaarheid te signaleren is de adres-status ADS van de 386DX ook ingeschakeld. Figuur 7/4.1-27 toont de snelste leescycli met gepijplijnde adres-timing.



Figuur 7/4.1-26: De kortst mogelijke leescycli met niet-gepijplijnde adres-timing.

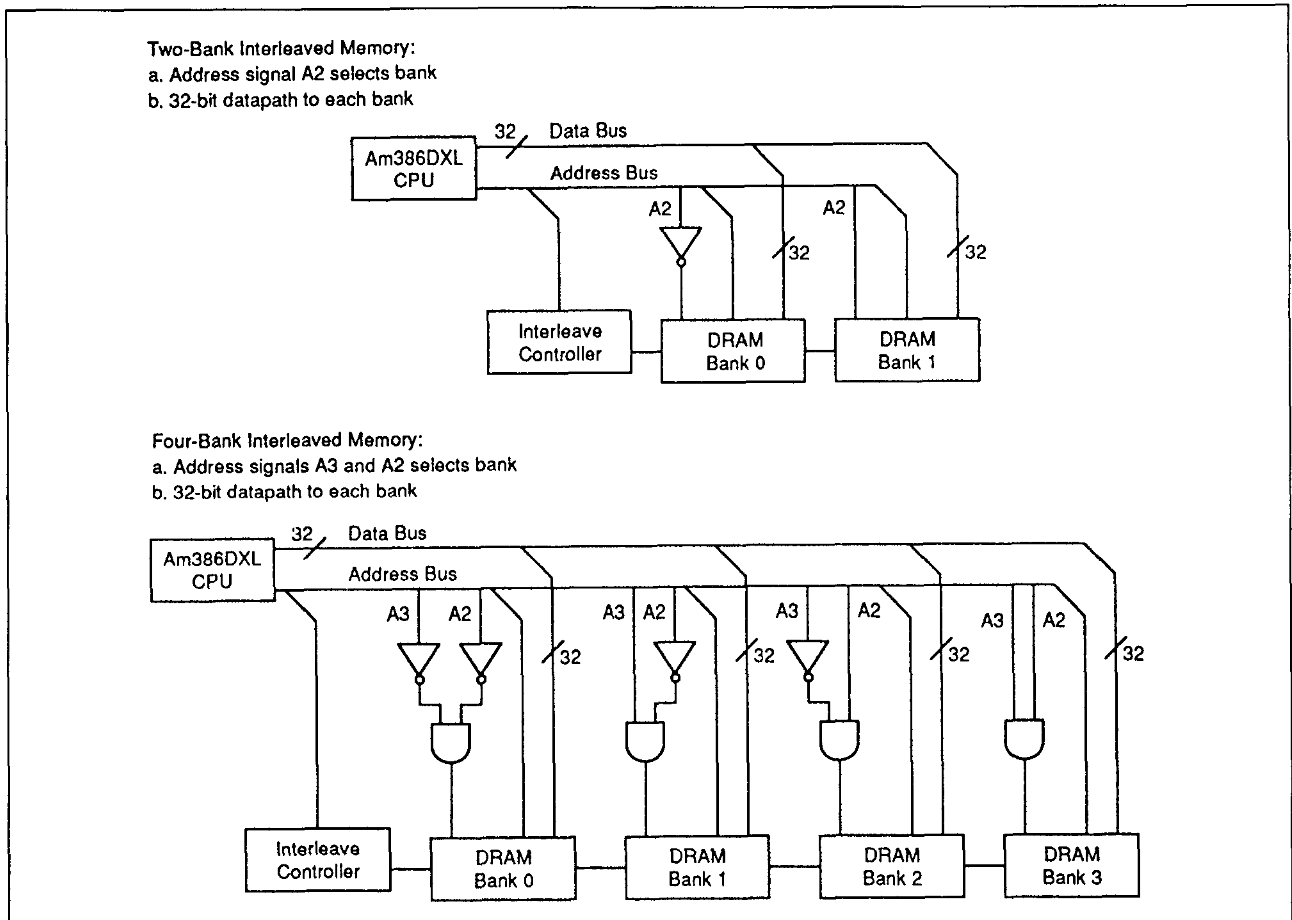
4.1 80386



Figuur 7/4.1-27: De kortst mogelijke leescycli met gepijplijnde adres-timing.

In figuur 7/4.1-27 is te zien dat de snelste buscycli die gepijplijnde adres-timing gebruiken slechts twee bus-toestanden (T1P en T2P) nodig hebben. Daarom is bij cycli met gepijplijnde adres-timing dezelfde databandbreedte mogelijk als bij niet-gepijplijnde cycli. Alleen is de adres-naar-data toegangstijd toegenomen ten opzichte van een niet-gepijplijnde cyclus.
Door de adres-naar-data toegangstijd te verlengen, nemen de aan de wait states gestelde eisen af bij gepijplijnde adres-timing. Als bijvoorbeeld één wait-state nodig is bij niet-gepijplijnde adres-timing, zijn er geen nodig bij gepijplijnde adres-timing.
Gepijplijnde adres-timing is nuttig in systemen die beschikken over adres-latches. In die systemen kan de decodeer-schakeling, zodra een adres is gelatcht, door de gepijplijnde beschikbaarheid van het volgende adres chip-selects (en andere noodzakelijke select-signalen) van te voren genereren. Hierdoor wordt onmiddellijk bij het begin van de volgende cyclus toegang verkregen tot de geselecteerde schakelingen. Met andere woorden: het einde van de lopende cyclus kan de decodeertijd voor de volgende cyclus overlappen.
Als een systeem een geheugen-structuur van twee of meer elkaar aanvullende ("interleaving") geheugenbanken bevat maakt gepijplijnde adres-timing nog meer overlappende aktiviteiten mogelijk. De interleaved memory-controller moet dan zo ontworpen zijn dat de volgende geheugen-operatie in de ene geheugenbank begint, terwijl de lopende buscyclus al een andere geheugenbank aktiveert. Figuur 7/4.1-28 laat de algemene structuur van de 386DX processor zien met 2-bank en 4-bank aanvullend geheugen.

4.1 80386



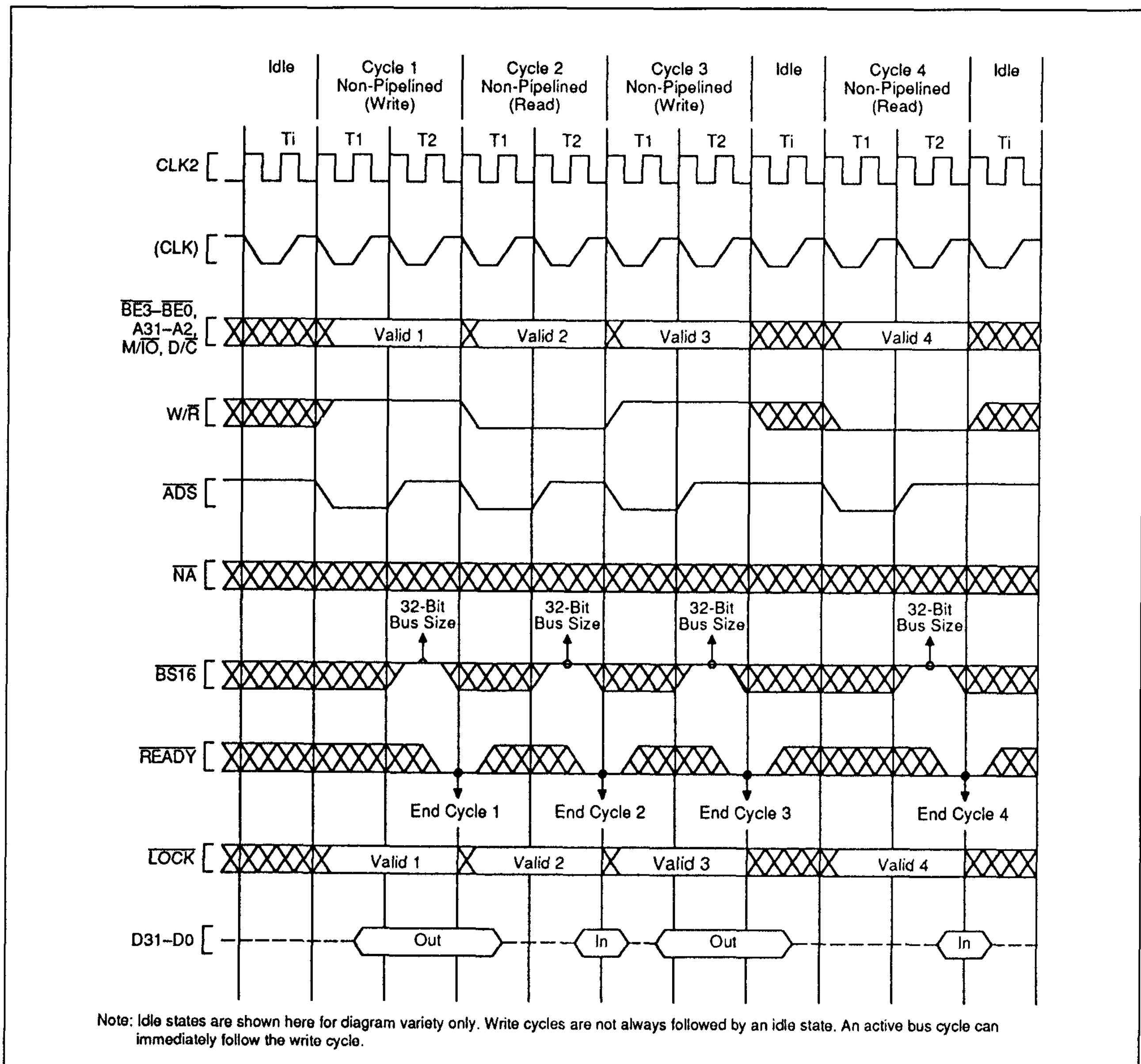
Figuur 7/4.1-28: Geheugenstructuur met 2-bank en 4-bank interleaved geheugen.

Lees- en schrijfcycli

Data-overdrachten ontstaan als gevolg van buscycli, die worden geclassificeerd als lees- of schrijfcycli. Bij leescycli wordt data overgebracht van een externe schakeling naar de processor, bij schrijfcycli is de richting andersom.

Twee soorten adres-timing kunnen dynamisch worden geselecteerd: niet-gepijplijnd of gepijplijnd. Na een vrijloop-toestand van de bus gebruikt de processor altijd niet-gepijplijnde adres-timing. De $\overline{NA}$-ingang (Next Address) kan echter worden ingeschakeld om voor de volgende cyclus gepijplijnde adres-timing te kiezen. Wanneer pijplijning is geselecteerd en de 386DX intern een busrequest heeft ontvangen, komen het adres en de definitie van de volgende cyclus al beschikbaar voordat de lopende buscyclus wordt bevestigd door $\overline{READY}$. Over het algemeen wordt de $\overline{NA}$-ingang iedere buscyclus bemonsterd om de gewenste adrestiming voor de volgende buscyclus te selecteren. Zoals al is geschreven kunnen twee fysieke databus-breedten dynamisch worden ingesteld: 32 bit en 16 bit. Over het algemeen wordt de $\overline{BS16}$-ingang aan het einde van de buscyclus bemonsterd om te bevestigen dat de fysieke databus-breedte overeenkomt met de lopende cyclus. Bij indicatie van een 16 bit bus ($\overline{BS16}$ aanwezig) voert de processor automatisch de overdracht uit op een 16 bit databus. Afhankelijk van de breedte en uitlijning van de operand kan nog een buscyclus nodig zijn, waarbij D0 tot en met D15 worden gebruikt in plaats van D16 tot en met D31 (zie tabel 7/4.1-21 voor de details).

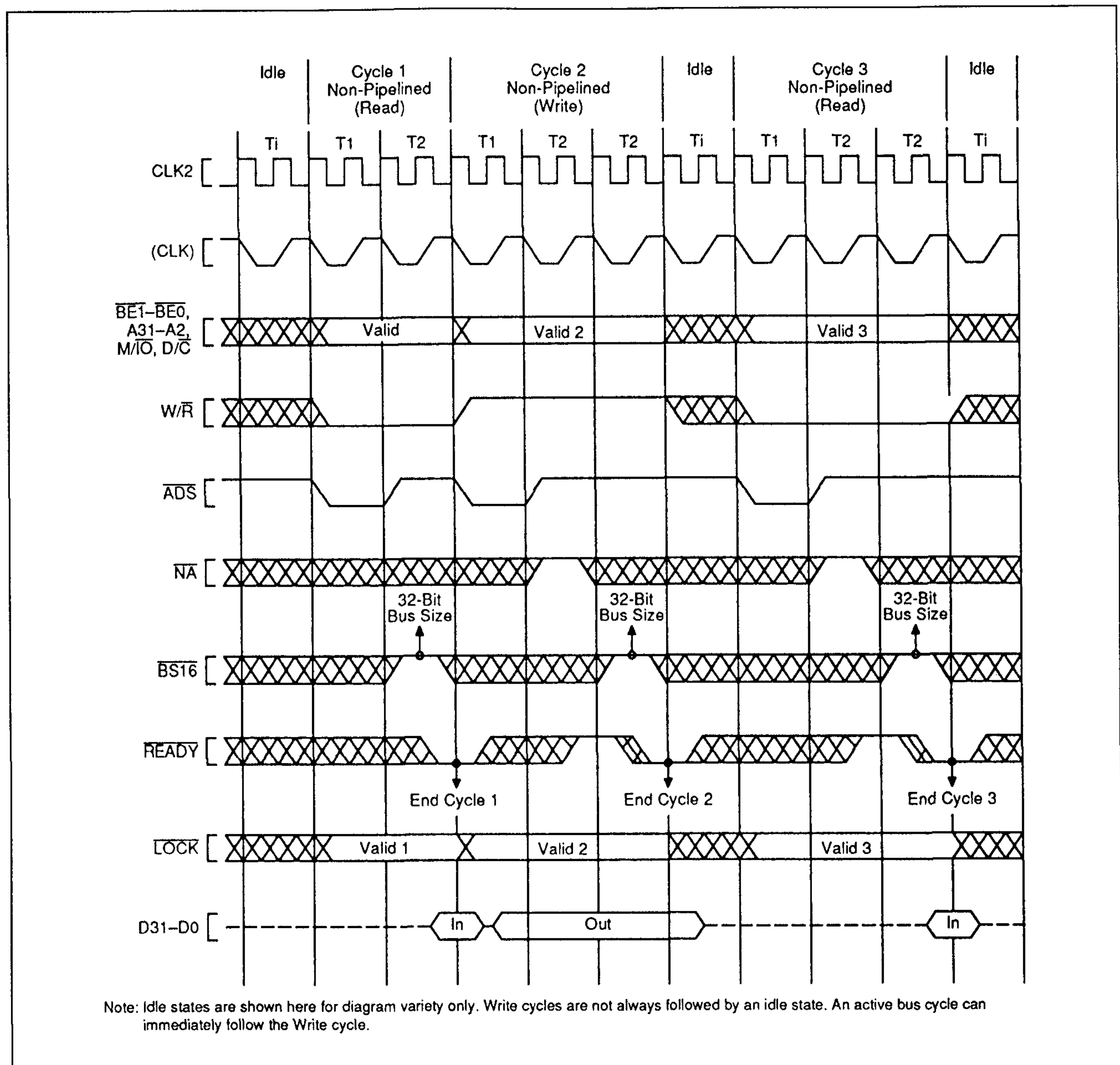
4.1 80386



Figuur 7/4.1-29: Verschillende buscycli en vrijloop-toestanden bij niet-gepijplijnde adressen (geen wait-states).

Voor het beëindigen van een lees- of schrijfcyclus is, net als bij elke andere buscyclus, bevestiging van de cyclus (acknowledge) door het READY-signaal nodig.
Zolang dat niet is verschenen, plaatst de processor wait-states in de buscyclus om aanpassing op de snelheid van de externe schakelingen mogelijk te maken. Aan het einde van de tweede bus-toestand binnen de buscyclus wordt READY bemonsterd. Als externe hardware op dat moment de buscyclus bevestigt door READY in te schakelen, eindigt de buscyclus zoals in figuur 7/4.1-29 te zien is. Als READY niet aanwezig is (zie figuur 7/4.1-30) gaat de cyclus nog een bustoestand door (een wait-state) en wordt READY aan het einde van die toestand nogmaals bemonsterd. Dit gaat onbegrensd door totdat de cyclus tenslotte met een READY wordt bevestigd.

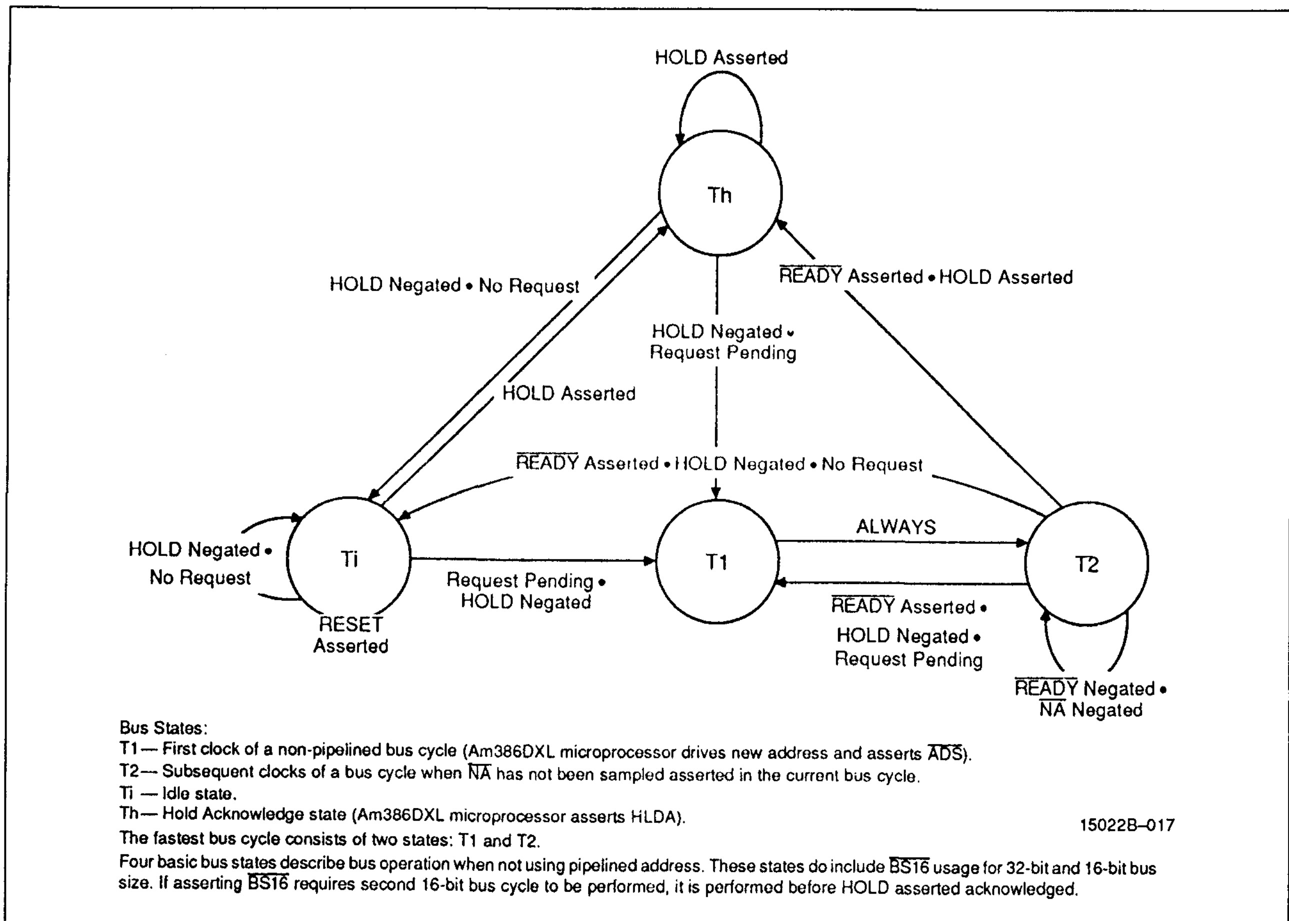
4.1 80386



Figuur 7/4.1-30: Verschillende buscycli en vrijloop-toestanden bij niet-gepijplijnde adressen (meerdere wait-states).

Wanneer de lopende buscyclus wordt bevestigd, beëindigt de 386DX processor die. Wordt een leescyclus bevestigd, dan latcht de 386DX de informatie die op de datapennen staat.
Bij bevestiging van een schrijfcyclus blijft de schrijf-data gedurende de gehele fase één van de volgende bus-toestand geldig om data hold-tijd te verschaffen.

Niet-gepijplijnd adres
Iedere buscyclus mag worden uitgevoerd met niet-gepijplijnde adres-timing. Figuur 7/4.1-29 bevat bijvoorbeeld verschillende lees- en schrijfcycli met niet-gepijplijnde adres-timing. Hierin zijn de snelst mogelijke cycli met niet-gepijplijnde adressen te zien die twee bus-toestanden per buscyclus hebben.

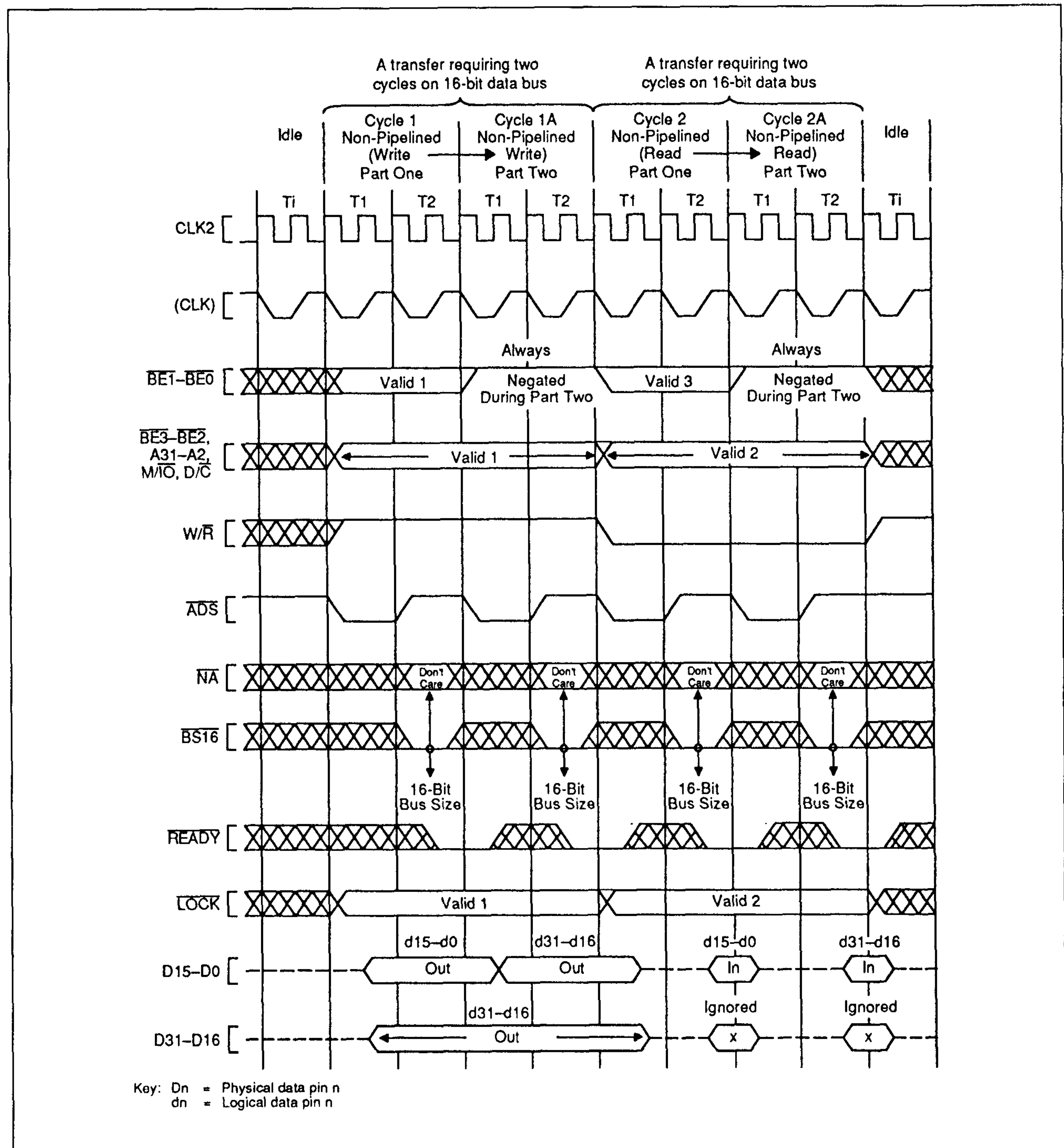


Figuur 7/4.1-31: De 386DX microprocessor bus-toestanden (wanneer geen gepijplijnde adressen worden gebruikt).

De toestanden worden T1 en T2 genoemd. In de fase van T1 worden de adres-signalen en buscyclus-definitiesignalen geldig en om de beschikbaarheid daarvan te signaleren wordt de adres-status (ADS) ingeschakeld. Bij een leescyclus laat de 386DX processor zijn data-signalen zweven, zodat ze door de externe schakeling kunnen worden aangedreven.

Aan het einde van iedere leescyclus moeten alle data-pennen op een geldig logisch niveau (HOOG of LAAG) zijn wanneer READY wordt ingeschakeld. Bij een schrijfcyclus worden de data-pennen door de 386DX aangedreven, beginnend op fase twee van T1 tot fase één van de bus-toestand die volgt op de bevestiging van de cyclus. In figuur 7/4.1-30 worden niet-gepijplijnde buscycli getoond, waarbij één wait aan de cycli 2 en 3 wordt toegevoegd. Aan het einde van de eerste T2 in de cycli 2 en 3 wordt READY niet aangetroffen, zodat T2 daarbij moet worden herhaald. Wanneer geen pijplijning wordt gebruikt, blijven het adres en de buscyclus-definitie geldig gedurende alle wait-states. Wanneer er wait-states worden toegevoegd en het gewenst is niet-gepijplijnde adres-timing te handhaven, is het nodig om NA gedurende elke T2-toestand weg te halen, behalve bij de laatste (zie figuur 7/4.1-30, cycli 2 en 3). Als NA bij bemonstering op een andere T2 dan de laatste ingeschakeld blijkt te zijn, is de volgende toestand T2I of T2P (beide voor gepijplijnd adres) in plaats van nog een T2 (voor niet-gepijplijnd adres).

4.1 80386



Figuur 7/4.1-32: Het inschakelen van $\overline{BS16}$ (geen wait-states, niet-gepijplijnd adres).

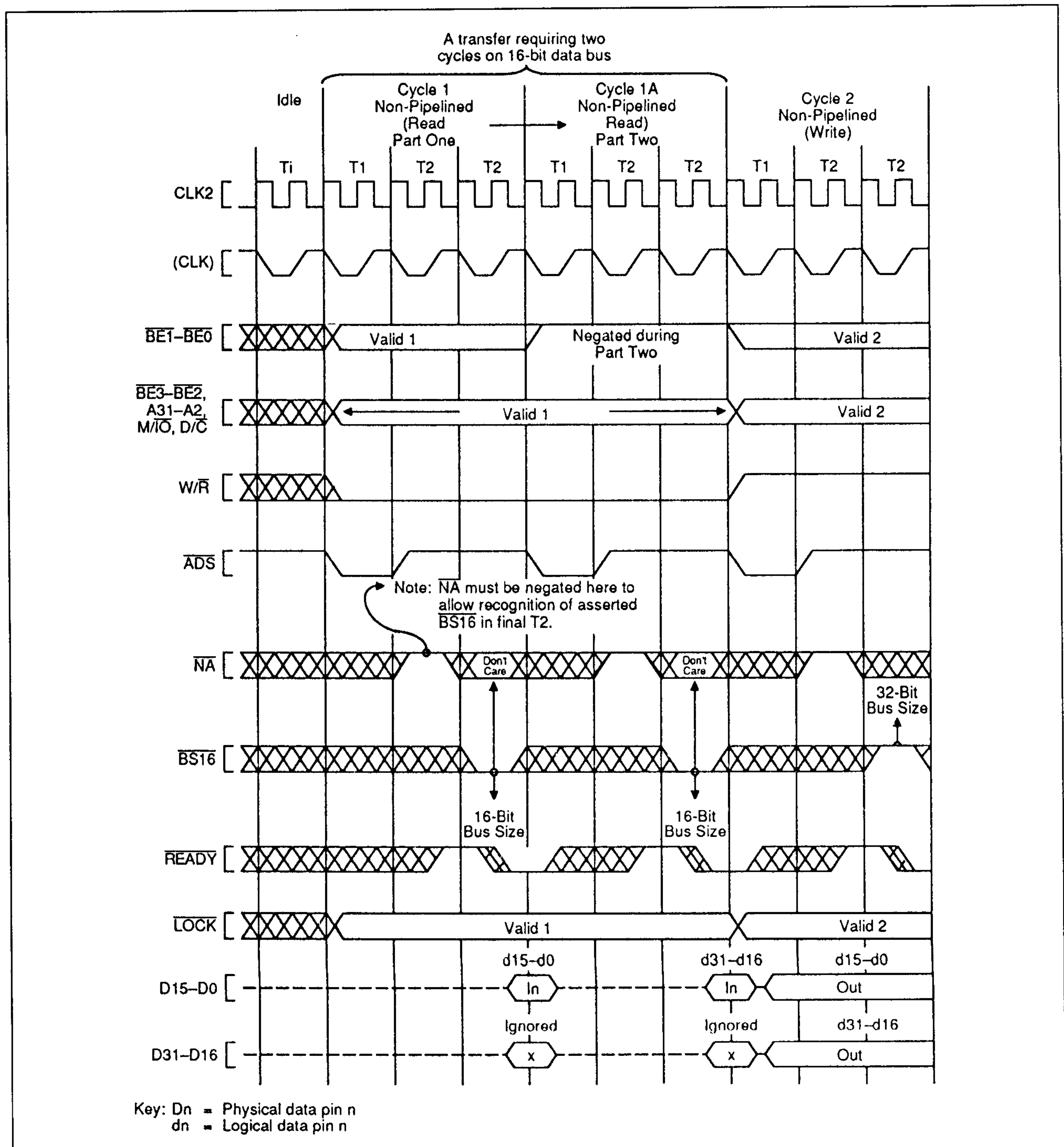
Wanneer geen pijplijning wordt gebruikt, worden de bus-toestanden en overgangen volledig geïllustreerd door figuur 7/4.1-31: de bus-overgangen tussen vier mogelijke toestanden T1, T2, Ti en Th. Buscycli bestaan uit T1 en T2, waarbij T2 wordt herhaald voor wait-states. Anders kan de bus vrijlopend, in de Ti-toestand of in de hold-toestand Th zijn. Wanneer de bus vrijlopend is (idle), bevindt hij zich in toestand Ti. Buscycli beginnen

altijd met T1, waarna T2 volgt. Als een buscyclus niet wordt bevestigd, wordt T2 herhaald. Wordt een cyclus gedurende T2 bevestigd, dan volgt T1 van de volgende buscyclus als intern een bus-request wacht, of Ti als er geen bus-request is geweest, of Th als de HOLD-ingang is ingeschakeld.
Het bus-statusdiagram (figuur 7/4.1-31) is ook geldig bij het gebruik van BS16, omdat de bus-afmeting geen invloed heeft op de externe bus-toestanden. Als een extra 16 bit buscyclus nodig is om de overdracht te voltooien, worden dezelfde toestand-overgangen gebruikt.

Niet-gepijplijnd adres met dynamische databus-breedteregeling
De fysieke databus-breedte voor een willekeurige niet-gepijplijnde buscyclus kan 32 bit of 16 bit zijn. Aan het begin van de buscyclus gedraagt de processor zich alsof de databus 32 bit breed is. Wanneer de buscyclus wordt bevestigd (door READY aan het eind van een T2-toestand) bepaalt de meest recente bemonstering van BS16 de databus-breedte van de zojuist bevestigde cyclus. Als BS16 is ingeschakeld (16 bit) en er twee 16 bit buscycli nodig zijn om de overdracht te voltooien, moet BS16 gedurende de tweede cyclus zijn ingeschakeld. Net als alle andere buscycli moet de tweede 16 bit cyclus worden bevestigd met READY.
Wanneer een tweede 16 bit buscyclus nodig is om de overdracht via een 16 bit bus te voltooien, staan de adressen die voor de twee 16 bit buscycli worden gegenereerd in nauw verband tot elkaar. De adressen zijn dezelfde, maar BE0 en BE1 worden altijd bij de tweede cyclus weggehaald. Dit wordt gedaan omdat de data op D0 tot en met D15 al bij de eerste 16 bit cyclus werd verstuurd. In de figuren 7/4.1-32 en -33 worden gevallen getoond waarbij het inschakelen van BS16 een tweede 16 bit cyclus noodzakelijk maakt (figuur 7/4.1-32 zonder wait-states en figuur -33 met één wait-state).
In figuur 7/4-1-33 wordt BS16 tijdens cyclus 1 ingeschakeld. Let op dat NA in de T2-toestand(en) vóór de laatste T2-toestand moet worden weggehaald. Dit is nodig om herkenning van een ingeschakelde BS16 in de laatste T2-toestand mogelijk te maken.

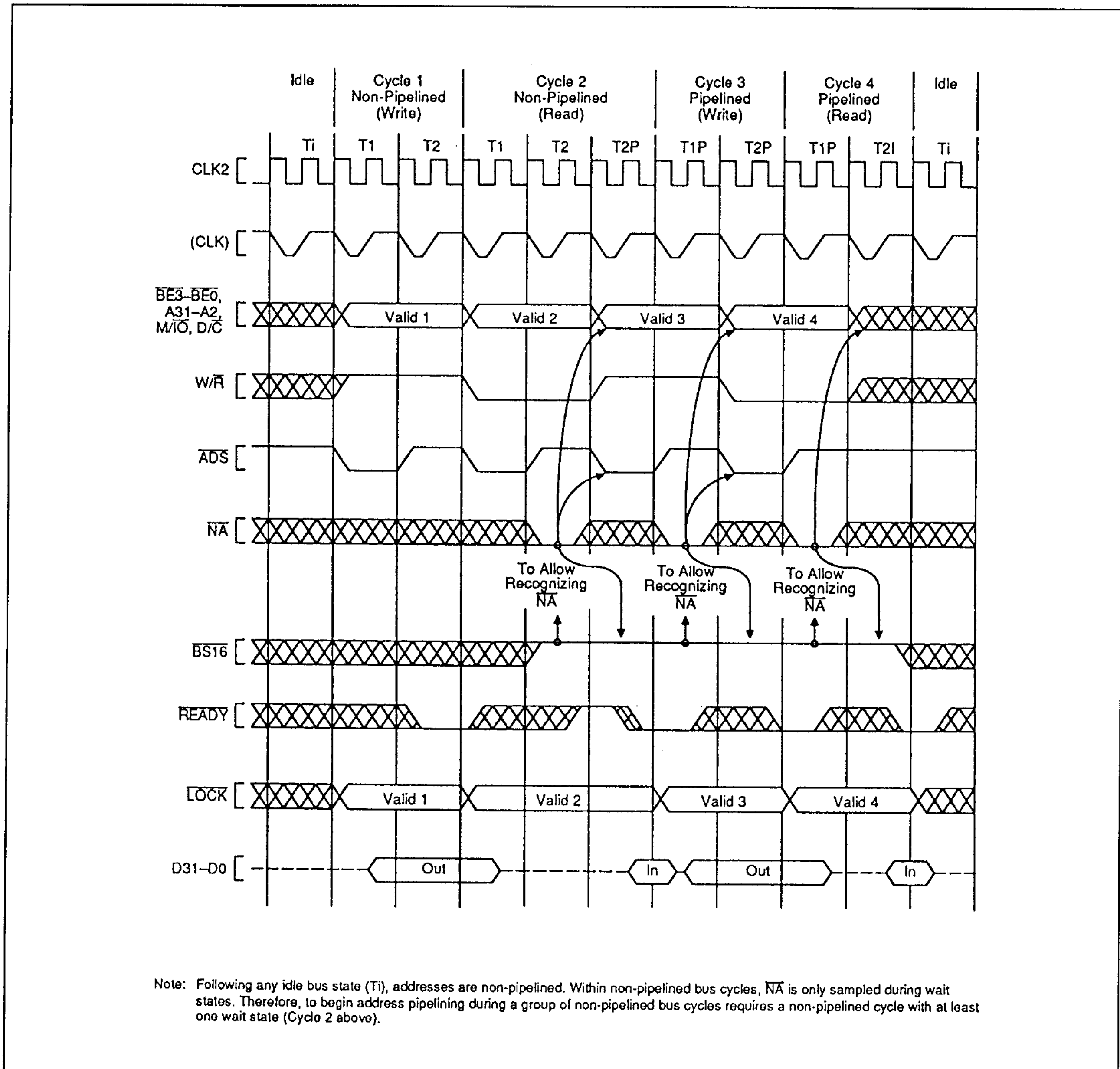
Gepijplijnd adres
Adres-pijplijning is de optie om het adres en de buscyclus-definitie van de volgende, intern hangende buscyclus aan te vragen voordat de lopende buscyclus wordt bevestigd met een READY. Wanneer het volgende adres wordt geleverd, schakelt de 386DX het ADS-signaal in. De adres-pijplijn optie wordt geregeld op een cyclus-voor-cyclus basis met het NA-signaal. Zodra een buscyclus aan de gang is en het huidige adres gedurende minimaal één gehele bus-toestand geldig is geweest, wordt de NA-ingang aan het einde van elke fase 1 bemonsterd totdat de buscyclus is bevestigd. Gedurende niet-gepijplijnde buscycli wordt NA daarom bemonsterd aan het einde van fase 1 in iedere T2. Een voorbeeld hiervan is cyclus 2 in figuur 7/4.1-34, waarbij NA aan het einde van fase 1 van elke T2 wordt bemonsterd (hij werd éénmaal ingeschakeld bij de eerste T2 en heeft gedurende die buscyclus geen effect meer). Indien NA ingeschakeld wordt aangetroffen is de 386DX processor vrij om het adres en de buscyclus-definitie van de volgende buscyclus uit te sturen en ADS in te schakelen zodra hij intern een bus-request heeft uitstaan. Het volgende adres mag reeds bij de volgende bustoestand verschijnen, of de lopende buscyclus nu bevestigd is of niet. Met betrekking tot de details van de adres-pijplijning heeft de 386DX microprocessor de volgende kenmerken:
- Om NA bij bemonstering ingeschakeld te vinden moet BS16 bij dat bemonsterende venster zijn weggehaald (zie figuur 7/4.1-34, de cycli 2 tot en met 4 en figuur 7/4.1-35, de cycli 1 tot en met 4). Als NA en BS16 bij de laatste T2-periode van een buscyclus ingeschakeld worden aangetroffen, heeft BS16 voorrang: de busbreedte wordt 16 bit en het volgende adres wordt niet gepijplijnd.

4.1 80386



Figuur 7/4.1-33: Het inschakelen van $\overline{BS16}$ (één wait-state, niet-gepijplijnd adres).

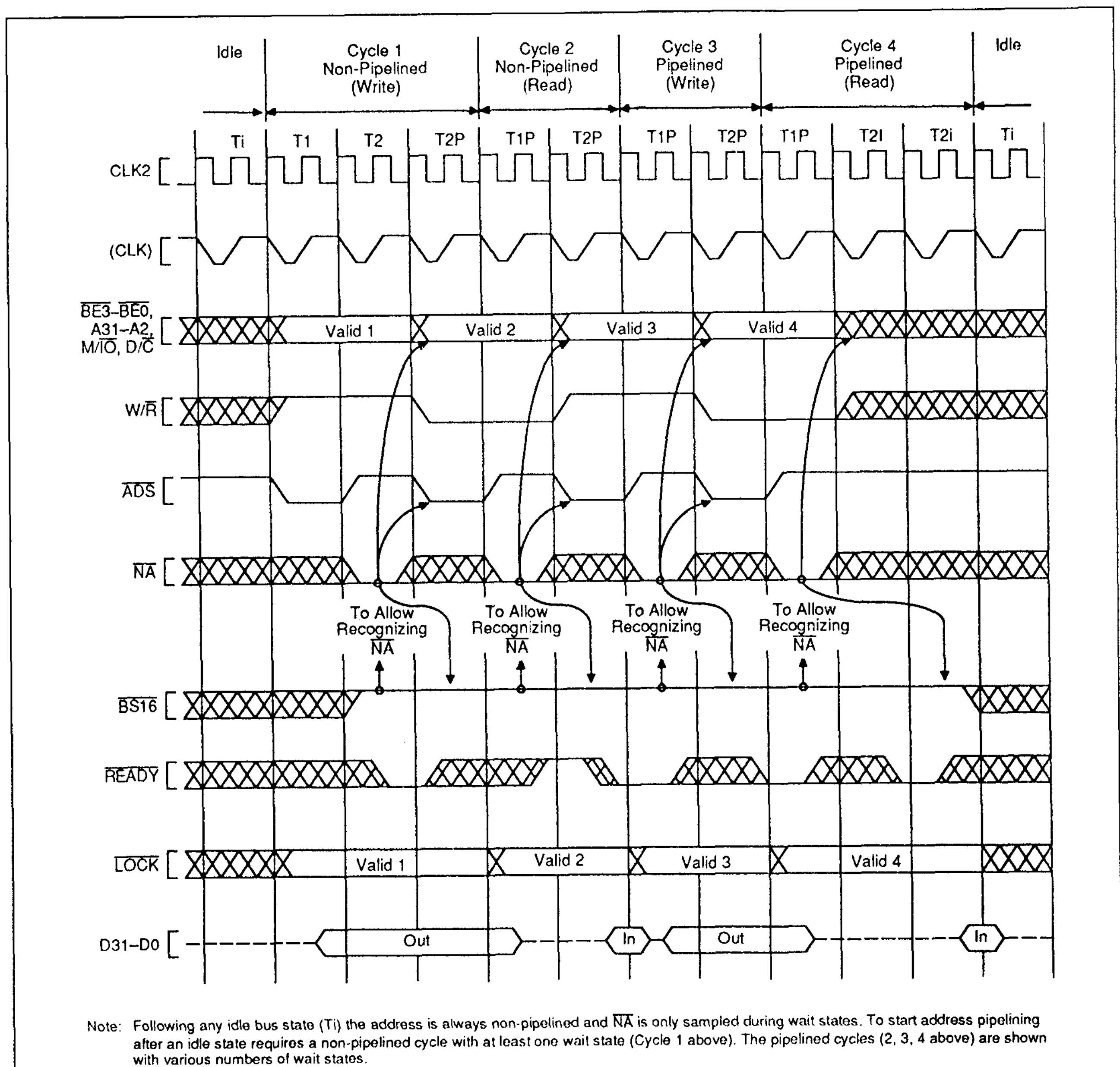
4.1 80386



Figuur 7/4.1-34: Het overgaan op gepijplijnd adres gedurende een reeks buscycli.

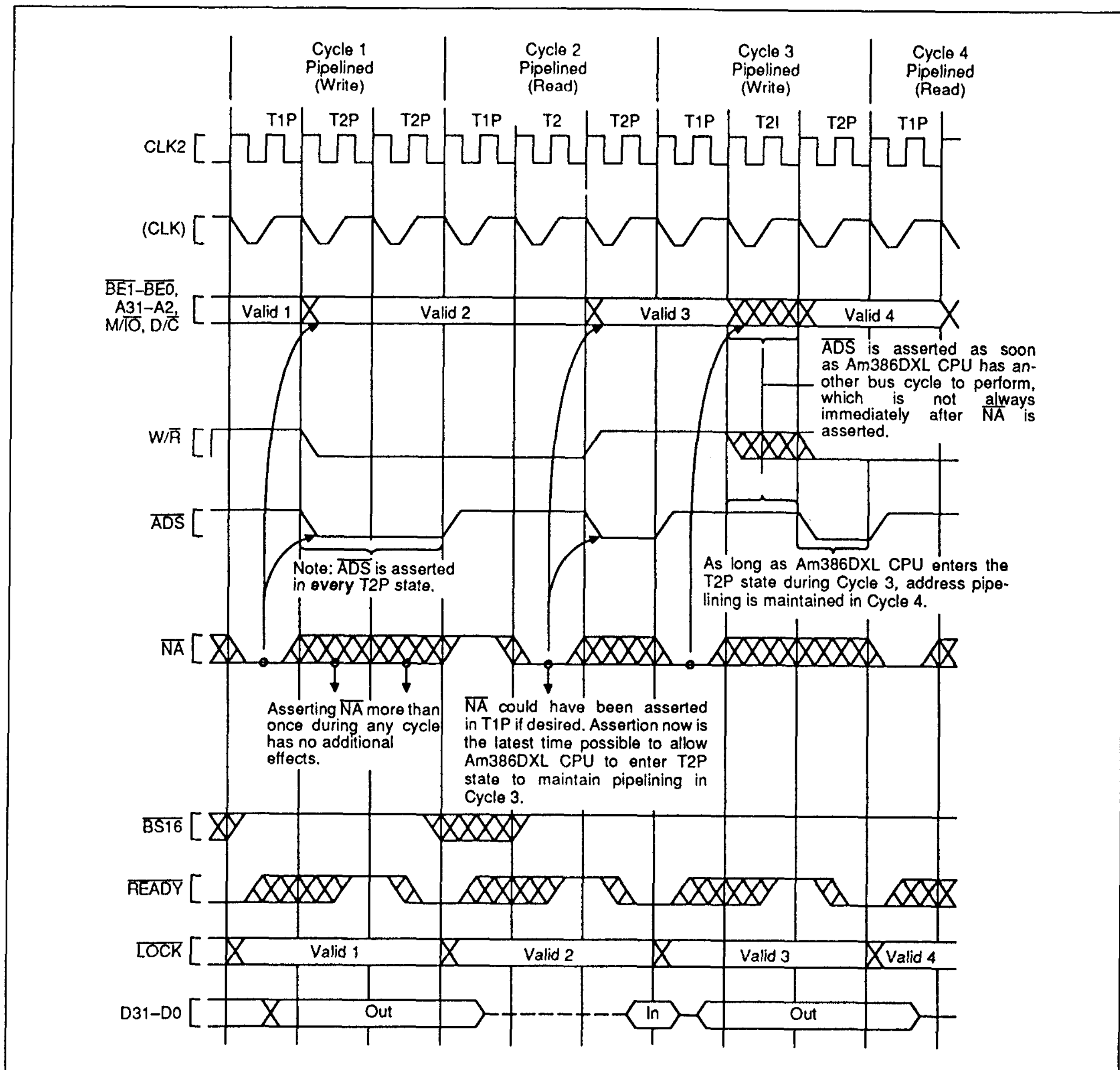
– Het volgende adres mag reeds verschijnen op de buscyclus nadat $\overline{NA}$ aanwezig werd gevonden (zie figuur 7/4.1-34 of -35). In dat geval wordt onmiddellijk overgegaan op toestand T2P. Wanneer echter niet reeds een intern bus-request hangt, komt het volgende adres niet direct na aantreffen van $\overline{NA}$ beschikbaar, waardoor T2i wordt binnengegaan in plaats van T2P (zie figuur 7/4.1-36, cyclus 3). Wanneer de lopende buscyclus nog niet door $\overline{READY}$ is bevestigd, wordt T2P betreden zodra de 386DX het volgende adres op de bus zet. Externe hardware moet daarom de $\overline{ADS}$-uitgang in de gaten houden ter bevestiging dat het volgende adres werkelijk op de bus staat.

4.1 80386



Figuur 7/4.1-35: Snelst mogelijke overgang op gepijplijnd adres na een vrijlopende bus-toestand.

- Zodra NA bij bemonstering ingeschakeld wordt aangetroffen begint de 386DX aan het bus-request met de hoogste prioriteit. Er kan geen extra 16 bit overdracht naar hetzelfde adres worden uitgevoerd in het geval dat BS16 extern zou zijn ingeschakeld. Er moet daarom worden aangenomen dat de huidige busbreedte 32 bit is. Als NA in een buscyclus ingeschakeld wordt aangetroffen, moet BS16 daarom later in die buscyclus worden weggehaald (zie de figuren 7/4.1-34, -35 en -36). Schakel NA dan ook niet in tijdens buscycli die een ingeschakelde BS16 nodig hebben.

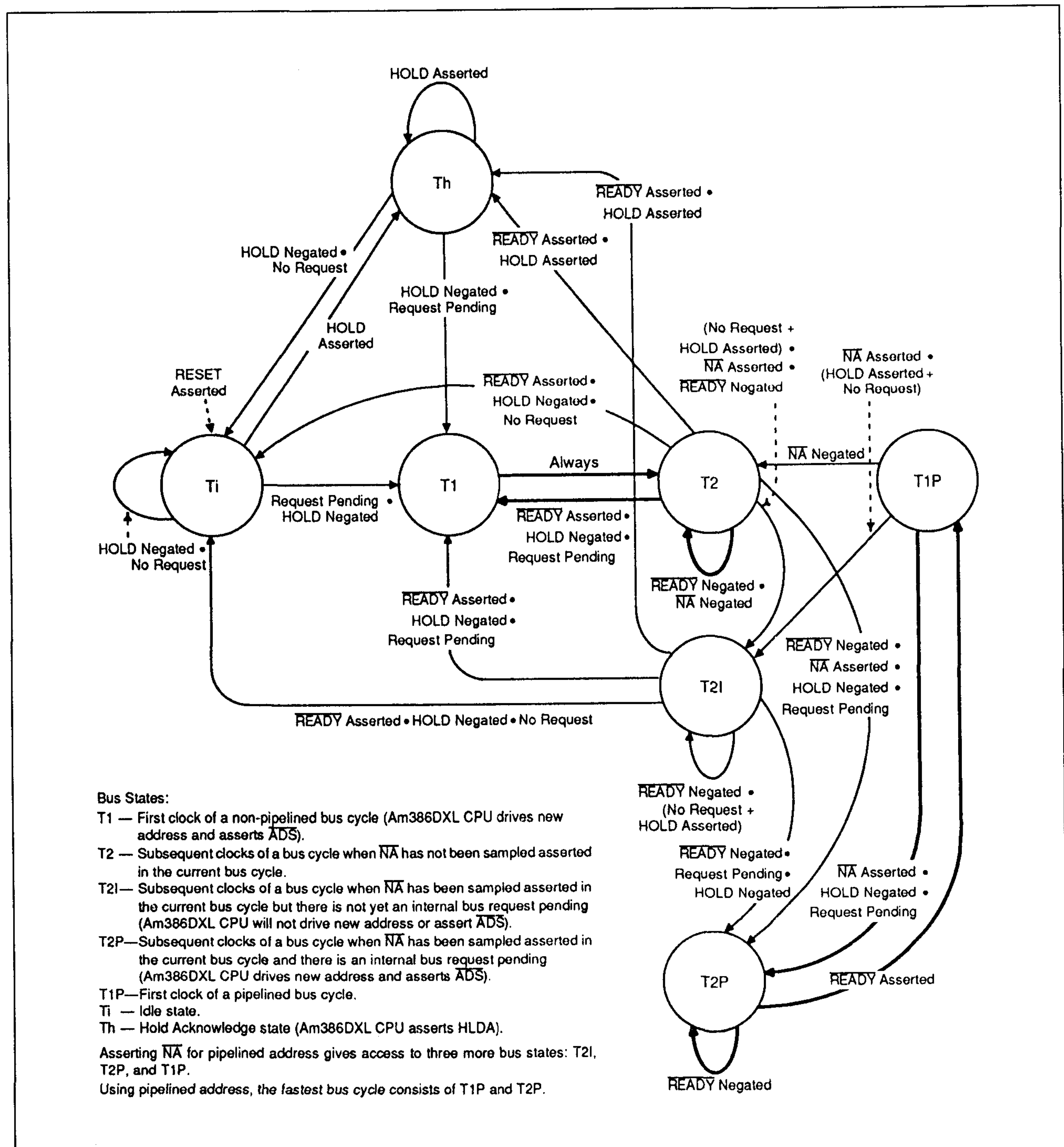


Figuur 7/4.1-36: Details van adres-pijplijning gedurende cycli met wait-states.

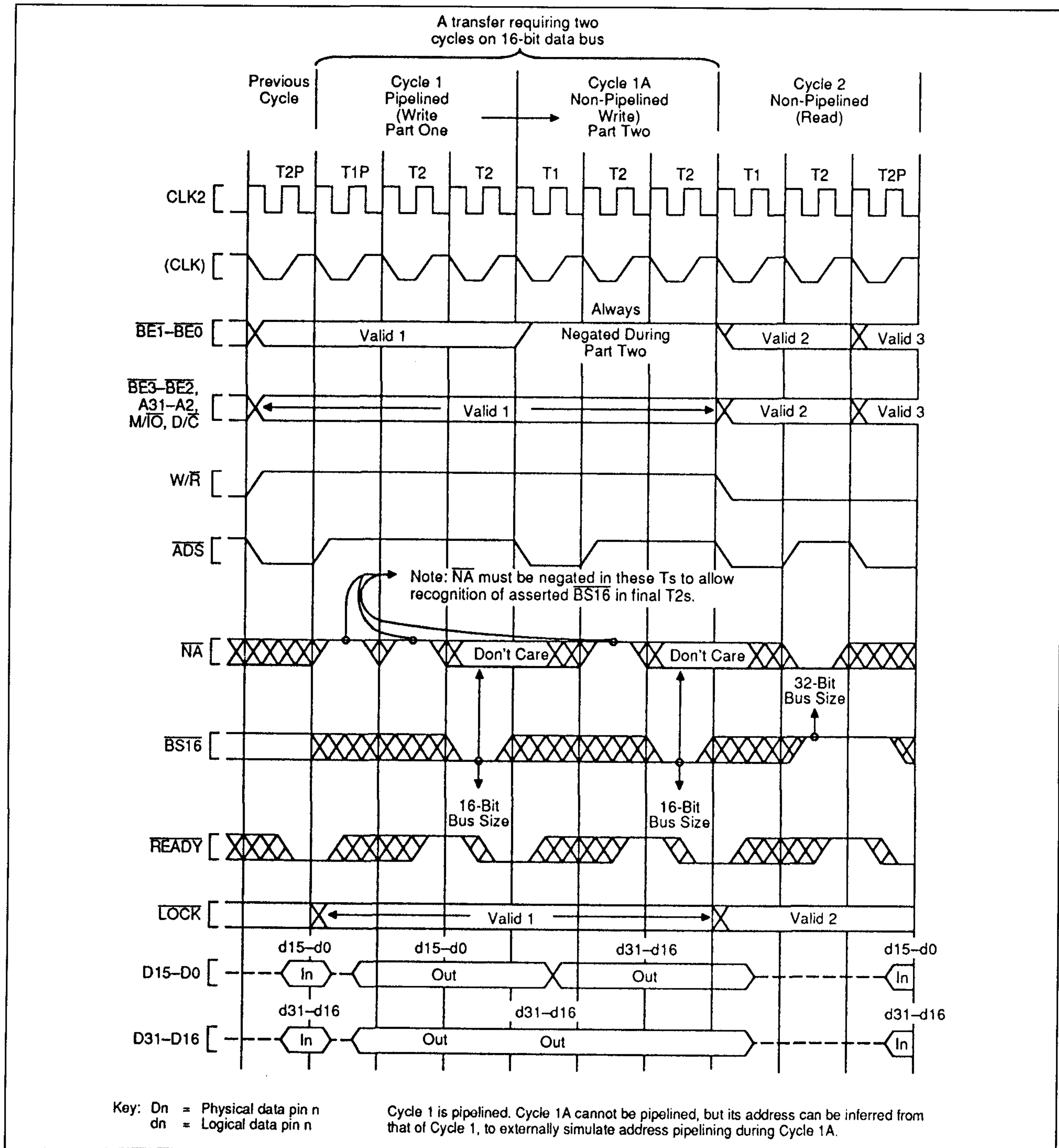
- Elk adres dat met een puls op de ADS-uitgang wordt geldig verklaard, blijft gedurende tenminste twee processor clock-perioden stabiel op de adrespennen. De 386DX kan niet vaker dan iedere twee clock-perioden een nieuw adres produceren (zie figuren 7/4.1-34, -35 en -36).
- Alleen het adres en de buscyclus-definitie van de allereerst komende buscyclus is beschikbaar. De pijplijn mogelijkheid kan niet verder dan één buscyclus vooruit kijken (figuur 7/4.1-36, cyclus 1).

Figuur 7/4.1-37 toont het complete overgangenschema van de bus-toestanden, inclusief de werking van gepijplijnde adressen. In

deze uitbreiding op figuur 7/4.1-31 zijn de drie extra bus-toestanden voor gepijplijnde adressen vet gedrukt. De snelste cyclus met een gepijplijnd adres bestaat uit slechts twee bus-toestanden, T1P en T2P.



Figuur 7/4.1-37: Alle 386DX microprocessor bus-toestanden (inclusief gepijplijnde adressen).



Figuur 7/4.1-38: Het gebruik van $\overline{NA}$ en $\overline{BS16}$.

Gepijplijnd adres met dynamische bus-breedte regeling

Het $\overline{BS16}$-signaal maakt ook de interfacing op 16 bit databussen eenvoudig mogelijk. Wanneer dit signaal aanwezig is, voert de 386DX bus-interface hardware de overdracht uit via D0 tot en met D15. Er bestaat echter een bepaalde mate van interactie tussen het gebruik van adres-pijplijning en gebruik van busbreedte-16. Dit wordt veroor-

zaakt doordat het overbrengen van 32 bit operands over een 16 bit bus meerdere buscycli nodig heeft. Als de overdracht beide 16 bit helften van de 32 bit bus beslaat, moet de 386DX daarvoor een tweede buscyclus uitvoeren, hetgeen tegen het gebruik van NA ingaat.
Wanneer NA ingeschakeld wordt gevonden, moet de 386DX de volgende, intern hangende bus-request bedienen en mag het volgende, intern hangende adres op de bus worden gezet. Het inschakelen van NA maakt het daarom onmogelijk voor de volgende buscyclus om weer toegang te krijgen tot het huidige adres op A2 tot en met A31 (wat door het inschakelen van BS16 door de externe hardware misschien gevraagd werd).

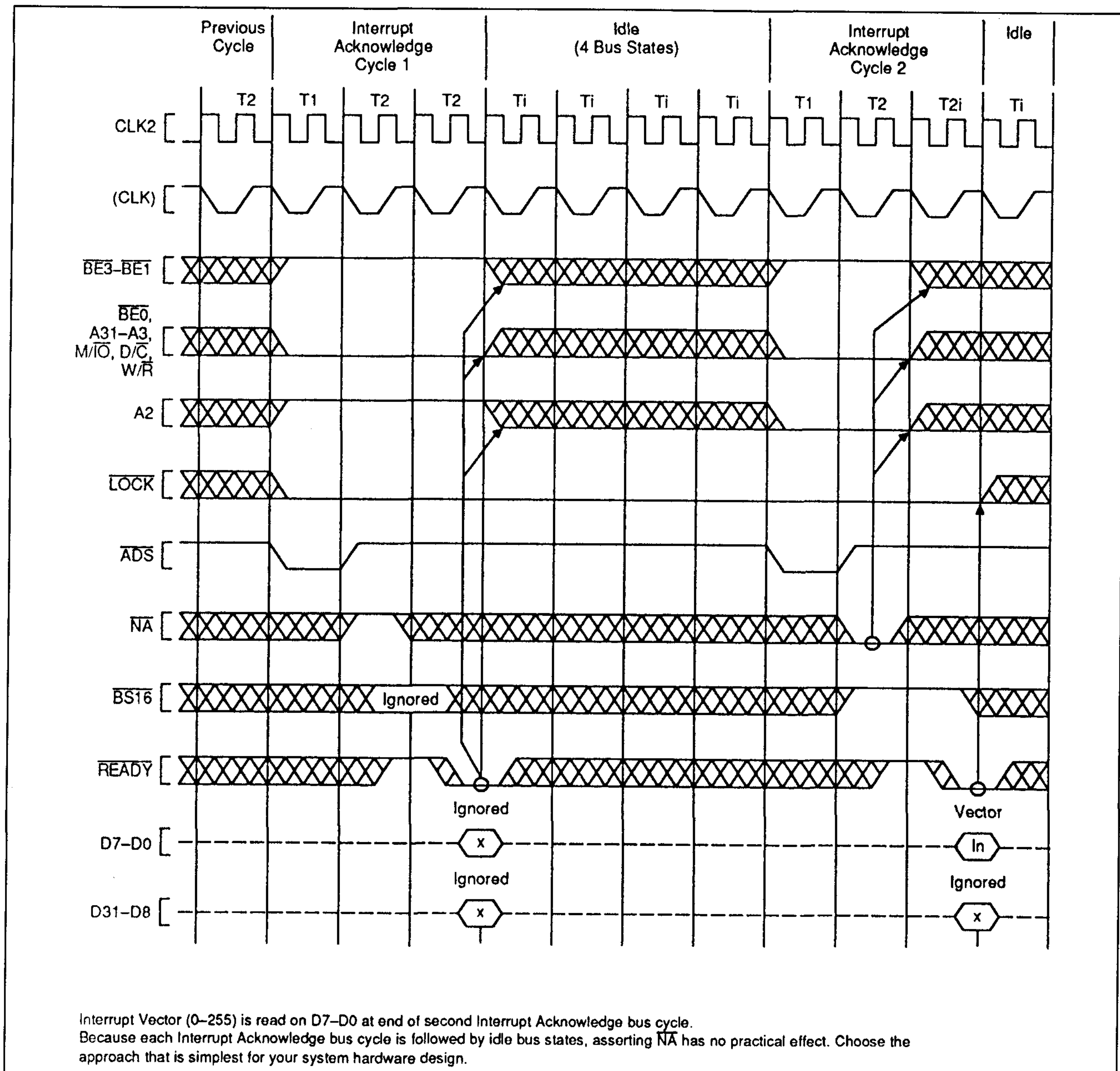
Conflict voorzieningen
Om conflicten te vermijden heeft de 386DX processor de volgende voorzieningen:
- BS16 moet in de lopende buscyclus worden weggehaald als bij bemonstering NA reeds was ingeschakeld. Van de huidige databus wordt aangenomen dat die 32 bit breed is.
- Als NA en BS16 in hetzelfde bemonsterende venster beide blijken te zijn ingeschakeld, heeft BS16 voorrang en doet de processor net alsof NA niet was ingeschakeld.

Sommige typen 16 bit of 8 bit operands hebben geen aanpassing nodig om correct te worden overgedragen via een 16 bit bus. Dat zijn lees- of schrijfoperands die alleen de laagste helft van de databus en schrijfoperands die slechts de hoogste helft van de bus gebruiken, omdat de 386DX gelijktijdig de schrijfdata op de laagste helft van de databus dupliceert. Voor deze patronen van Byte Enables en de R/W-signalen behoeft BS16 niet te worden ingeschakeld, zodat NA desgewenst tijdens de buscyclus aanwezig kan zijn.

Interrupt Acknowledge (INTA) cycli
Als antwoord op een interrupt request op de INTR-ingang (wanneer interrupts zijn vrijgegeven) voert de 386DX processor twee bevestigingscycli (interrupt acknowledge) uit. Deze buscycli lijken in zoverre op leescycli dat bus-definitiesignalen het soort busaktiviteit bepalen en dat elke cyclus doorgaat totdat die wordt bevestigd met READY.
Met de toestand van A2 wordt onderscheid gemaakt tussen de eerste en de tweede interrupt acknowledge cyclus. Het byteadres bij de eerste cyclus is 4 (A3 tot en met A31 = LAAG, A2 = HOOG, BE1 tot en met BE3 = HOOG en BE0 = LAAG) en bij de tweede interrupt acknowledge cyclus is het 0 (A2 dan ook LAAG). Zoals in figuur 7/4.1-39 te zien is, is de LOCK-uitgang ingeschakeld vanaf het begin van de eerste interrupt acknowledge cyclus tot aan het eind van de tweede. De 386DX voegt vier vrijloop bustoestanden Ti toe tussen beide interrupt acknowledge cycli om compatibel te zijn met de TRHRL-specificatie van de 8259A Interrupt Controller.
Gedurende beide interrupt acknowledge cycli zweven D0 tot en met D31. Aan het einde van de eerste interrupt acknowledge cyclus wordt geen data gelezen, maar aan het eind van de tweede leest de 386DX een externe interrupt-vector in van D0 tot en met D7. De vector geeft het specifieke interrupt-nummer (tussen 0 en 255) dat bediend moet worden.

Halt-indicatie cyclus
De 386DX stopt als gevolg van het uitvoeren van een HALT-instructie. Bij het binnengaan van de halt-toestand wordt een halt-indicatie-cyclus uitgevoerd. De halt-indicatie-cyclus is herkenbaar aan de toestand van de bus-definitie signalen en een byte-adres van 2. BE0 en BE2 zijn de enige signalen die onderscheid maken tussen een halt-indicatie en een shutdown-indicatie (met een adres van 0). Gedurende de halt-cyclus komt ongedefinieerde data op D0 tot en met D31 te staan. De halt-indicatie-cyclus moet worden bevestigd door READY. Een gestopte 386DX processor hervat de werkzaamheden wanneer INTR (bij vrijgegeven interrupts), NMI of RESET wordt gegeven.

4.1 80386

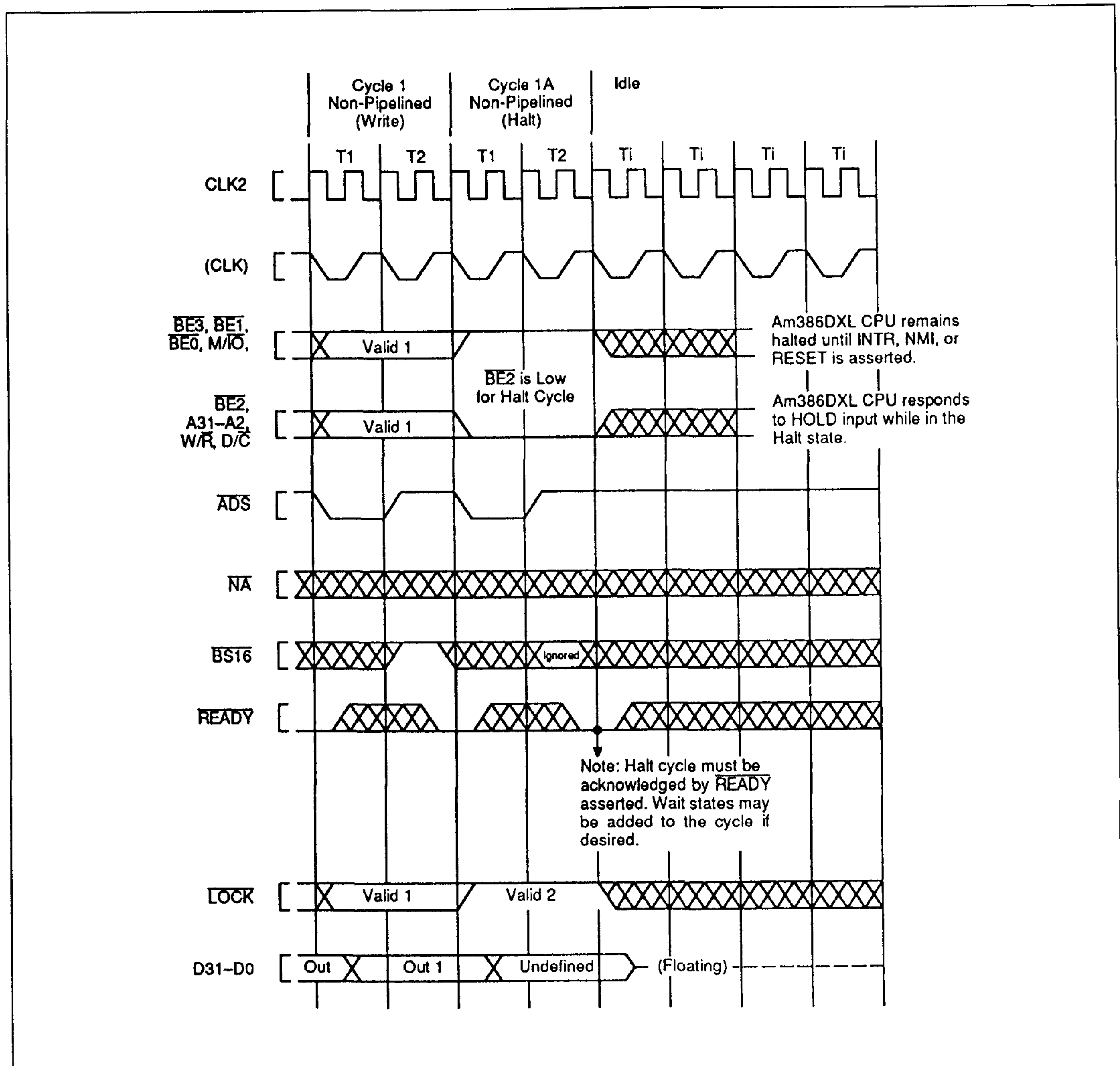


Figuur 7/4.1-39: Interrupt Acknowledge-cycli.

Shutdown-indicatie cyclus

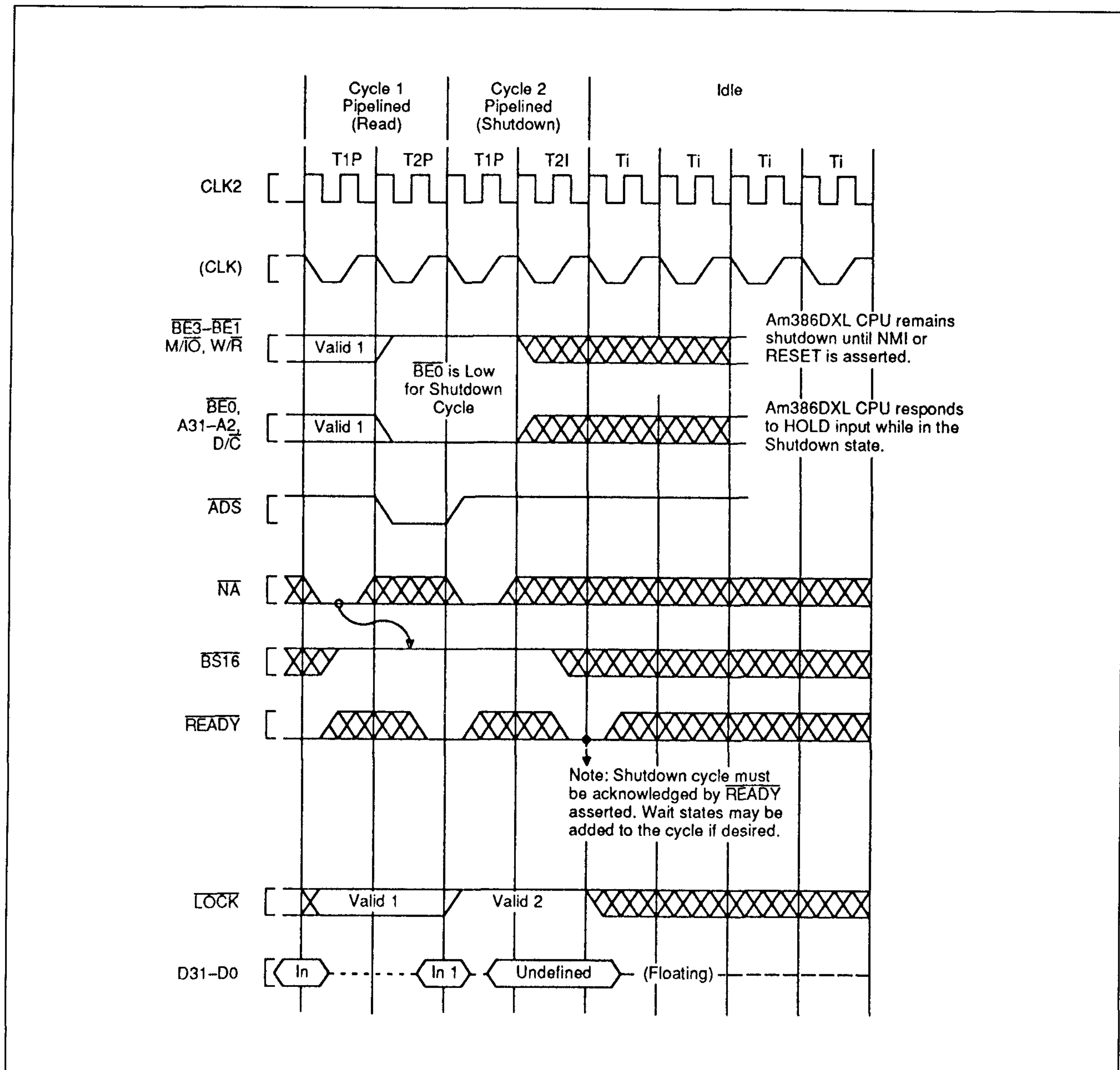
De 386DX wordt stopgezet als gevolg van een beveiligingsfout wanneer wordt geprobeerd een dubbele fout uit te voeren. Bij het binnengaan van de shutdown-toestand wordt dan een shutdown-indicatie-cyclus uitgevoerd. Deze is herkenbaar aan de toestand van de bus-definitie signalen en een byte-adres van 0. Alleen met $\overline{BE0}$ en $\overline{BE2}$ kan onderscheid worden gemaakt tussen een shutdown-indicatie en een halt-indicatie (met een adres van 2). Ook tijdens de shutdown-cyclus komt ongedefinieerde data op D0 tot en met D31. De shutdown-indicatie-cyclus moet worden bevestigd door $\overline{READY}$. Een met shutdown gestopte 386DX processor gaat pas door wanneer NMI of RESET wordt gegeven.

4.1 80386



Figuur 7/4.1-40: Een HALT indicatie-cyclus.

4.1 80386



Figuur 7/4.1-41: Een Shutdown indicatie-cyclus.

Bespreking van de instructieset

Inleiding

In dit gedeelte wordt de instructieset van de 386DX microprocessor beschreven. In tabel 7/4.1-23a tot en met -23n worden alle instructies, inclusief coderingen en benodigde clock-cycli opgesomd.

Daarna volgen verdere details van de instructie-codering waarbij de structuur van de codering en de definities van de velden worden behandeld. Om de tijdsduur van een instructie te berekenen moet het aantal clock-cycli uit tabel 7/4.1-23 worden vermenigvuldigd met de processor clock-periode (bijvoorbeeld 50 ns voor een 20 MHz 386DX processor, of 30 ns voor een 33 MHz type).

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
GENERAL DATA TRANSFER						
MOV = Move:						
Register to Register/Memory	1 0 0 0 1 0 0 w | mod reg r/m		2/2	2/2	b	h
Register/Memory to Register	1 0 0 0 1 0 1 w | mod reg r/m		2/4	2/4	b	h
Immediate to Register/Memory	1 1 0 0 0 1 1 w | mod 0 0 0 r/m | immediate data		2/2	2/2	b	h
Immediate to Register (short form)	1 0 1 1 w reg | immediate data		2	2		
Memory to Accumulator (short form)	1 0 1 0 0 0 0 w | full displacement		4	4	b	h
Accumulator to Memory (short form)	1 0 1 0 0 0 1 w | full displacement		2	2	b	h
Register Memory to Segment Register	1 0 0 0 1 1 1 0 | mod sreg3 r/m		2/5	18/19	b	h, i, j
Segment Register to Register/Memory	1 0 0 0 1 1 0 0 | mod sreg3 r/m		2/2	2/2	b	h
MOVSX = Move With Sign Extension						
Register From Register/Memory	0 0 0 0 1 1 1 1 | 1 0 1 1 1 1 1 w | mod reg r/m		3/6	3/6	b	h
MOVZX = Move With Zero Extension						
Register From Register/Memory	0 0 0 0 1 1 1 1 | 1 0 1 1 0 1 1 w | mod reg r/m		3/6	3/6	b	h
PUSH = Push:						
Register/Memory	1 1 1 1 1 1 1 1 | mod 1 1 0 r/m		5	5	b	h
Register (short form)	0 1 0 1 0 reg		2	2	b	h
Segment Register (ES, CS, SS or DS)	0 0 0 sreg2 1 1 0		2	2	b	h
Segment Register (FS or GS)	0 0 0 0 1 1 1 1 | 1 0 sreg3 0 0 0		2	2	b	h
Immediate	0 1 1 0 1 0 s 0 | immediate data		2	2	b	h
PUSHA = Push All	0 1 1 0 0 0 0 0		18	18	b	h
POP = Pop						
Register/Memory	1 0 0 0 1 1 1 1 | mod 0 0 0 r/m		5	5	b	h
Register (short form)	0 1 0 1 1 reg		4	4	b	h
Segment Register (ES, SS or DS)	0 0 0 sreg 2 1 1 1		7	21	b	h, i, j
Segment Register (FS or GS)	0 0 0 0 1 1 1 1 | 1 0 sreg 3 0 0 1		7	21	b	h, i, j
POPA = Pop All	0 1 1 0 0 0 0 1		24	24	b	h
XCHG = Exchange						
Register/Memory With Register	1 0 0 0 0 1 1 w | mod reg r/m		3/5	3/5	b, f	f, h
Register With Accumulator (short form)	1 0 0 1 0 reg		3	3		
IN = Input from:						
Fixed Port	1 1 1 0 0 1 0 w | port number	†26	12	6*/26**		m
Variable Port	1 1 1 0 1 1 0 w	†27	13	7*/27**		m
OUT = Output to:						
Fixed Port	1 1 1 0 0 1 1 w | port number	†24	10	4*/24**		m
Variable Port	1 1 1 0 1 1 1 w	†25	11	5*/25**		m
LEA = Load EA to Register	1 0 0 0 1 1 0 1 | mod reg r/m		2	2		

* If CPL ≤ IOPL ** If CPL > IOPL

Tabel 7/4.1-23a: Samenvatting van de 386DX instructieset, deel 1.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT				CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
SEGMENT CONTROL								
LDS = Load Pointer to DS	11000101	mod reg r/m			7	22	b	h, i, j
LES = Load Pointer to ES	11000100	mod reg r/m			7	22	b	h, i, j
LFS = Load Pointer to FS	00001111	10110100	mod reg r/m		7	25	b	h, i, j
LGS = Load Pointer to GS	00001111	10110101	mod reg r/m		7	25	b	h, i, j
LSS = Load Pointer to SS	00001111	10110010	mod reg r/m		7	22	b	h, i, j
FLAG CONTROL								
CLC = Clear Carry Flag	11111000				2	2		
CLD = Clear Direction Flag	11111100				2	2		
CLI = Clear Interrupt Enable Flag	11111010				8	8		m
CLTS = Clear Task Switched Flag	00001111	00000110			6	6	c	l
CMC = Complement Carry Flag	11110101				2	2		
LAHF = Load AH into Flag	10011111				2	2		
POPF = Pop Flags	10011101				5	5	b	h, n
PUSHF = Push Flags	10011100				4	4	b	h
SAHF = Store AH into Flags	10011110				3	3		
STC = Set Carry Flag	11111001				2	2		
STD = Set Direction Flag	11111101				2	2		
STI = Set Interrupt Enable Flag	11111011				8	8		m
ARITHMETIC								
ADD = Add								
Register to Register	000000dw	mod reg r/m			2	2		
Register to Memory	0000000w	mod reg r/m			7	7	b	h
Memory to Register	0000001w	mod reg r/m			6	6	b	h
Immediate to Register/Memory	100000sw	mod 000 r/m	immediate data		2/7	2/7	b	h
Immediate to Accumulator (short form)	0000010w	immediate data			2	2		
ADC = Add With Carry								
Register to Register	000100dw	mod reg r/m			2	2		
Register to Memory	0001000w	mod reg r/m			7	7	b	h
Memory to Register	0001001w	mod reg r/m			6	6	b	h
Immediate to Register/Memory	100000sw	mod 010 r/m	immediate data		2/7	2/7	b	h
Immediate to Accumulator (short form)	0001010w	immediate data			2	2		
INC = Increment								
Register/Memory	1111111w	mod 000 r/m			2/6	2/6	b	h
Register (short form)	01000 reg				2	2		
SUB = Subtract								
Register from Register	001010dw	mod reg r/m			2	2		

Tabel 7/4.1-23b: Samenvatting van de 386DX instructieset, deel 2.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
ARITHMETIC (Continued)					
Register from Memory	0010100w | mod reg r/m	7	7	b	h
Memory from Register	0010101w | mod reg r/m	6	6	b	h
Immediate from Register/Memory	100000sw | mod 101 r/m | immediate data	2/7	2/7	b	h
Immediate from Accumulator (short form)	0010110w | immediate data	2	2		
SBB = Subtract with Borrow					
Register from Register	000110dw | mod reg r/m	2	2		
Register from Memory	0001100w | mod reg r/m	7	7	b	h
Memory from Register	0001101w | mod reg r/m	6	6	b	h
Immediate from Register/Memory	100000sw | mod 011 r/m | immediate data	2/7	2/7	b	h
Immediate from Accumulator (short form)	0001110w | immediate data	2	2		
DEC = Decrement					
Register/Memory	1111111w | reg 001 r/m	2/6	2/6	b	h
Register (short form)	01001 reg	2	2		
CMP = Compare					
Register with Register	001110dw | mod reg r/m	2	2		
Memory with Register	0011100w | mod reg r/m	5	5	b	h
Register with Memory	0011101w | mod reg r/m	6	6	b	h
Immediate with Register/Memory	100000sw | mod 111 r/m | immediate data	2/5	2/5	b	h
Immediate with Accumulator (short form)	0011110w | immediate data	2	2		
NEG = Change Sign	1111011w | mod 011 r/m	2/6	2/6	b	h
AAA = ASCII Adjust for Add	00110111	4	4		
AAS = ASCII Adjust for Subtract	00111111	4	4		
DAA = Decimal Adjust for Add	00100111	4	4		
DAS = Decimal Adjust for Subtract	00101111	4	4		
MUL = Multiply (unsigned)					
Accumulator with Register/Memory	1111011w | mod 100 r/m				
Multiplier-Byte		12-17/15-20	12-17/15-20	b, d	d, h
-Word		12-25/15-28	12-25/15-28	b, d	d, h
-Doubleword		12-41/15-44	12-41/15-44	b, d	d, h
IMUL = Integer Multiply (signed)					
Accumulator with Register/Memory	1111011w | mod 101 r/m				
Multiplier-Byte		12-17/15-20	12-17/15-20	b, d	d, h
-Word		12-25/15-28	12-25/15-28	b, d	d, h
-Doubleword		12-41/15-44	12-41/15-44	b, d	d, h
Register with Register/Memory	00001111 | 10101111 | mod reg r/m				
Multiplier-Byte		12-17/15-20	12-17/15-20	b, d	d, h
-Word		12-25/15-28	12-25/15-28	b, d	d, h
-Doubleword		12-41/15-44	12-41/15-44	b, d	d, h
Register/Memory with Immediate to Register	011010s1 | mod reg r/m | immediate data				
-Word		13-26/14-27	13-26/14-27	b, d	d, h
-Doubleword		13-42/14-43	13-42/14-43	b, d	d, h

Tabel 7/4.1-23c: Samenvatting van de 386DX instructieset, deel 3.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT Real Address Mode or Virtual 8086 Mode	CLOCK COUNT Protected Virtual Address Mode	NOTES Real Address Mode or Virtual 8086 Mode	NOTES Protected Virtual Address Mode
ARITHMETIC (Continued)					
DIV = Divide (Unsigned)					
Accumulator by Register/Memory	1 1 1 1 0 1 1 w | mod 1 1 0 r/m				
Divisor—Byte		14/17	14/17	b,e	e,h
—Word		22/25	22/25	b,e	e,h
—Doubleword		38/41	38/41	b,e	e,h
IDIV = Integer Divide (Signed)					
Accumulator By Register/Memory	1 1 1 1 0 1 1 w | mod 1 1 1 r/m				
Divisor—Byte		19/22	19/22	b,e	e,h
—Word		27/30	27/30	b,e	e,h
—Doubleword		43/46	43/46	b,e	e,h
AAD = ASCII Adjust for Divide	1 1 0 1 0 1 0 1 | 0 0 0 0 1 0 1 0	19	19		
AAM = ASCII Adjust for Multiply	1 1 0 1 0 1 0 0 | 0 0 0 0 1 0 1 0	17	17		
CBW = Convert Byte to Word	1 0 0 1 1 0 0 0	3	3		
CWD = Convert Word to Double Word	1 0 0 1 1 0 0 1	2	2		
LOGIC					
Shift Rotate Instructions					
Not Through Carry **(ROL, ROR, SAL, SAR, SHL, and SHR)**					
Register/Memory by 1	1 1 0 1 0 0 0 w | mod TTT r/m	3/7	3/7	b	h
Register/Memory by CL	1 1 0 1 0 0 1 w | mod TTT r/m	3/7	3/7	b	h
Register/Memory by Immediate Count	1 1 0 0 0 0 0 w | mod TTT r/m | immed 8-bit data	3/7	3/7	b	h
Through Carry **(RCL and RCR)**					
Register/Memory by 1	1 1 0 1 0 0 0 w | mod TTT r/m	9/10	9/10	b	h
Register/Memory by CL	1 1 0 1 0 0 1 w | mod TTT r/m	9/10	9/10	b	h
Register/Memory by Immediate Count	1 1 0 0 0 0 0 w | mod TTT r/m | immed 8-bit data	9/10	9/10	b	h
	TTT Instruction 0 0 0 ROL 0 0 1 ROR 0 1 0 RCL 0 1 1 RCR 1 0 0 SHL/SAL 1 0 1 SHR 1 1 1 SAR				
SHLD = Shift Left Double					
Register/Memory by Immediate	0 0 0 0 1 1 1 1 | 1 0 1 0 0 1 0 0 | mod reg r/m | immed 8-bit data	3/7	3/7		
Register/Memory by CL	0 0 0 0 1 1 1 1 | 1 0 1 0 0 1 0 1 | mod reg r/m	3/7	3/7		
SHRD - Shift Right Double					
Register/Memory by Immediate	0 0 0 0 1 1 1 1 | 1 0 1 0 1 1 0 0 | mod reg r/m | immed 8-bit data	3/7	3/7		
Register/Memory by CL	0 0 0 0 1 1 1 1 | 1 0 1 0 1 1 0 1 | mod reg r/m	3/7	3/7		
AND = And					
Register to Register	0 0 1 0 0 0 d w | mod reg r/m	2	2		

Tabel 7/4.1-23d: Samenvatting van de 386DX instructieset, deel 4.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
LOGIC (Continued)						
Register to Memory	0010000w · mod reg r/m		7	7	b	h
Memory to Register	0010001w · mod reg r/m		6	6	b	h
Immediate to Register/Memory	1000000w · mod 100 r/m · immediate data		2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0010010w · immediate data		2	2		
TEST = And Function to Flags, No Result						
Register/Memory and Register	1000010w · mod reg r/m		2/5	2/5	b	h
Immediate Data and Register/Memory	1111011w · mod 000 r/m · immediate data		2/5	2/5	b	h
Immediate Data and Accumulator (Short Form)	1010100w · immediate data		2	2		
OR = Or						
Register to Register	000010dw · mod reg r/m		2	2		
Register to Memory	0000100w · mod reg r/m		7	7	b	h
Memory to Register	0000101w · mod reg r/m		6	6	b	h
Immediate to Register/Memory	1000000w · mod 001 r/m · immediate data		2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0000110w · immediate data		2	2		
XOR = Exclusive Or						
Register to Register	001100dw · mod reg r/m		2	2		
Register to Memory	0011000w · mod reg r/m		7	7	b	h
Memory to Register	0011001w · mod reg r/m		6	6	b	h
Immediate to Register/Memory	1000000w · mod 110 r/m · immediate data		2/7	2/7	b	h
Immediate to Accumulator (Short Form)	0011010w · immediate data		2	2		
NOT = Invert Register/Memory	1111011w · mod 010 r/m		2/6	2/6	b	h
STRING MANIPULATION						
CMPS = Compare Byte Word	1010011w		10	10	b	h
INS = Input Byte/Word from DX Port	0110110w	†29	15	9*/29**	b	h, m
LODS = Load Byte/Word to AL/AX/EAX	1010110w		5	5	b	h
MOVS = Move Byte Word	1010010w		8	8	b	h
OUTS = Output Byte/Word to DX Port	0110111w	†28	14	8*/28**	b	h, m
SCAS = Scan Byte Word	1010111w		8	8	b	h
STOS = Store Byte/Word from AL/AX/EX	1010101w		5	5	b	h
XLAT = Translate String	11010111		5	5		h
REPEATED STRING MANIPULATION Repeated by Count in CX or ECX						
REPE CMPS = Compare String (Find Non-Match)	11110011 · 1010011w		5+9n	5+9n	b	h

* If CPL ≤ IOPL ** If CPL > IOPL

Tabel 7/4.1-23e: Samenvatting van de 386DX instructieset, deel 5.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	Clk Count Virtual 8086 Mode	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
REPEATED STRING MANIPULATION (Continued)						
REPNE CMPS = Compare String						
(Find Match)	11110010 | 1010011w		5+9n	5+9n	b	h
REP INS = Input String	11110010 | 0110110w	†28+6n	14+6n	8+6n*/28+6n**	b	h, m
REP LODS = Load String	11110010 | 1010110w		5+6n	5+6n	b	h
REP MOVS = Move String	11110010 | 1010010w		8+4n	8+4n	b	h
REP OUTS = Output String	11110010 | 0110111w	†26+5n	12+5n	6+5n*/26+5n**	b	h, m
REPE SCAS = Scan String						
(Find Non-AL/AX/EAX)	11110011 | 1010111w		5+8n	5+8n	b	h
REPNE SCAS = Scan String						
(Find AL/AX/EAX)	11110010 | 1010111w		5+8n	5+8n	b	h
REP STOS = Store String	11110010 | 1010101w		5+5n	5+5n	b	h
BIT MANIPULATION						
BSF = Scan Bit Forward	00001111 | 10111100 | mod reg r/m		11+3n	11+3n	b	h
BSR = Scan Bit Reverse	00001111 | 10111101 | mod reg r/m		9+3n	9+3n	b	h
BT = Test Bit						
Register/Memory, Immediate	00001111 | 10111010 | mod 100 r/m | immed 8-bit data		3/6	3/6	b	h
Register/Memory, Register	00001111 | 10100011 | mod reg r/m		3/12	3/12	b	h
BTC = Test Bit and Complement						
Register/Memory, Immediate	00001111 | 10111010 | mod 111 r/m | immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 | 10111011 | mod reg r/m		6/13	6/13	b	h
BTR = Test Bit and Reset						
Register/Memory, Immediate	00001111 | 10111010 | mod 110 r/m | immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 | 10110011 | mod reg r/m		6/13	6/13	b	h
BTS = Test Bit and Set						
Register/Memory, Immediate	00001111 | 10111010 | mod 101 r/m | immed 8-bit data		6/8	6/8	b	h
Register/Memory, Register	00001111 | 10101011 | mod reg r/m		6/13	6/13	b	h
CONTROL TRANSFER						
CALL = Call						
Direct Within Segment	11101000 | full displacement		7+m	7+m	b	r
Register/Memory						
Indirect Within Segment	11111111 | mod 010 r/m		7+m/ 10+m	7+m/ 10+m	b	h, r
Direct Intersegment	10011010 | *unsigned full offset, selector*		17+m	34+m	b	j,k,r

NOTES:

† Clock count shown applies if I/O permission allows I/O to the port in virtual 8086 mode. If I/O bit map denies permission exception 13 fault occurs; refer to clock counts for INT 3 instruction.

* If CPL ≤ IOPL ** If CPL > IOPL

Tabel 7/4.1-23f: Samenvatting van de 386DX instructieset, deel 6.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)					
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			52 + m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			86 + m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			94 + 4x + m		h,j,k,r
From 80286 Task to 80286 TSS			273		h,j,k,r
From 80286 Task to 386DX™ CPU TSS			298		h,j,k,r
From 80286 Task to Virtual 8086 Task (386DX™ CPU TSS)			218		h,j,k,r
From 386DX™ CPU Task to 80286 TSS			273		h,j,k,r
From 386DX™ CPU Task to 386DX™ CPU TSS			300		h,j,k,r
From 386DX™ CPU Task to Virtual 8086 Task (386DX™ CPU TSS)			218		h,j,k,r
Indirect Intersegment	1 1 1 1 1 1 1 1 | mod 0 1 1 r/m	22 + m	38 + m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			56 + m		h,j,k,r
Via Call Gate to Different Privilege Level, (No Parameters)			90 + m		h,j,k,r
Via Call Gate to Different Privilege Level, (x Parameters)			98 + 4x + m		h,j,k,r
From 80286 Task to 80286 TSS			278		h,j,k,r
From 80286 Task to 386DX™ CPU TSS			303		h,j,k,r
From 80286 Task to Virtual 8086 Task (386DX™ CPU TSS)			222		h,j,k,r
From 386DX™ CPU Task to 80286 TSS			278		h,j,k,r
From 386DX™ CPU Task to 386DX™ CPU TSS			305		h,j,k,r
From 386DX™ CPU Task to Virtual 8086 Task (386DX™ CPU TSS)			222		h,j,k,r
JMP = Unconditional Jump					
Short	1 1 1 0 1 0 1 1 | 8-bit displacement	7 + m	7 + m		r
Direct within Segment	1 1 1 0 1 0 0 1 | full displacement	7 + m	7 + m		r
Register/Memory Indirect within Segment	1 1 1 1 1 1 1 1 | mod 1 0 0 r/m	7 + m/ 10 + m	7 + m/ 10 + m	b	h,r
Direct Intersegment	1 1 1 0 1 0 1 0 | unsigned full offset, selector	12 + m	27 + m		j,k,r
Protected Mode Only (Direct Intersegment)					
Via Call Gate to Same Privilege Level			45 + m		h,j,k,r
From 80286 Task to 80286 TSS			274		h,j,k,r
From 80286 Task to 386DX™ CPU TSS			301		h,j,k,r
From 80286 Task to Virtual 8086 Task (386DX™ CPU TSS)			219		h,j,k,r
From 386DX™ CPU Task to 80286 TSS			270		h,j,k,r
From 386DX™ CPU Task to 386DX™ CPU TSS			303		h,j,k,r
From 386DX™ CPU Task to Virtual 8086 Task (386DX™ CPU TSS)			221		h,j,k,r
Indirect Intersegment	1 1 1 1 1 1 1 1 | mod 1 0 1 r/m	17 + m	31 + m	b	h,j,k,r
Protected Mode Only (Indirect Intersegment)					
Via Call Gate to Same Privilege Level			49 + m		h,j,k,r
From 80286 Task to 80286 TSS			279		h,j,k,r
From 80286 Task to 386DX™ CPU TSS			306		h,j,k,r
From 80286 Task to Virtual 8086 Task (386DX™ CPU TSS)			223		h,j,k,r
From 386DX™ CPU Task to 80286 TSS			275		h,j,k,r
From 386DX™ CPU Task to 386DX™ CPU TSS			308		h,j,k,r
From 386DX™ CPU Task to Virtual 8086 Task (386DX™ CPU TSS)			225		h,j,k,r

Tabel 7/4.1-23g: Samenvatting van de 386DX instructieset, deel 7.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT			CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONTROL TRANSFER (Continued)							
RET = Return from CALL:							
Within Segment	1 1 0 0 0 0 1 1			10 + m	10 + m	b	g, h, r
Within Segment Adding Immediate to SP	1 1 0 0 0 0 1 0	16-bit displ		10 + m	10 + m	b	g, h, r
Intersegment	1 1 0 0 1 0 1 1			18 + m	32 + m	b	g, h, j, k, r
Intersegment Adding Immediate to SP	1 1 0 0 1 0 1 0	16-bit displ		18 + m	32 + m	b	g, h, j, k, r
Protected Mode Only (RET):							
to Different Privilege Level							
Intersegment					69		h, j, k, r
Intersegment Adding Immediate to SP					69		h, j, k, r
CONDITIONAL JUMPS							
NOTE: Times Are Jump "Taken or Not Taken"							
JO = Jump on Overflow							
8-Bit Displacement	0 1 1 1 0 0 0 0	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 0 0 0	full displacement	7 + m or 3	7 + m or 3		r
JNO = Jump on Not Overflow							
8-Bit Displacement	0 1 1 1 0 0 0 1	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 0 0 1	full displacement	7 + m or 3	7 + m or 3		r
JB/JNAE = Jump on Below/Not Above or Equal							
8-Bit Displacement	0 1 1 1 0 0 1 0	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 0 1 0	full displacement	7 + m or 3	7 + m or 3		r
JNB/JAE = Jump on Not Below/Above or Equal							
8-Bit Displacement	0 1 1 1 0 0 1 1	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 0 1 1	full displacement	7 + m or 3	7 + m or 3		r
JE/JZ = Jump on Equal/Zero							
8-Bit Displacement	0 1 1 1 0 1 0 0	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 0 0	full displacement	7 + m or 3	7 + m or 3		r
JNE/JNZ = Jump on Not Equal/Not Zero							
8-Bit Displacement	0 1 1 1 0 1 0 1	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 0 1	full displacement	7 + m or 3	7 + m or 3		r
JBE/JNA = Jump on Below or Equal/Not Above							
8-Bit Displacement	0 1 1 1 0 1 1 0	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 1 0	full displacement	7 + m or 3	7 + m or 3		r
JNBE/JA = Jump on Not Below or Equal/Above							
8-Bit Displacement	0 1 1 1 0 1 1 1	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 0 1 1 1	full displacement	7 + m or 3	7 + m or 3		r
JS = Jump on Sign							
8-Bit Displacement	0 1 1 1 1 0 0 0	8-bit displ		7 + m or 3	7 + m or 3		r
Full Displacement	0 0 0 0 1 1 1 1	1 0 0 0 1 0 0 0	full displacement	7 + m or 3	7 + m or 3		r

Tabel 7/4.1-23h: Samenvatting van de 386DX instructieset, deel 8.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT				CLOCK COUNT Real Address Mode or Virtual 8086 Mode	CLOCK COUNT Protected Virtual Address Mode	NOTES Real Address Mode or Virtual 8086 Mode	NOTES Protected Virtual Address Mode
CONDITIONAL JUMPS (Continued)								
JNS = Jump on Not Sign								
8-Bit Displacement	01111001	8-bit displ			7 + m or 3	7 + m or 3		r
Full Displacement	00001111	10001001	full displacement		7 + m or 3	7 + m or 3		r
JP/JPE = Jump on Parity/Parity Even								
8-Bit Displacement	01111010	8-bit displ			7 + m or 3	7 + m or 3		r
Full Displacement	00001111	10001010	full displacement		7 + m or 3	7 + m or 3		r
JNP/JPO = Jump on Not Parity/Parity Odd								
8-Bit Displacement	01111011	8-bit displ			7 + m or 3	7 + m or 3		r
Full Displacement	00001111	10001011	full displacement		7 + m or 3	7 + m or 3		r
JL/JNGE = Jump on Less/Not Greater or Equal								
8-Bit Displacement	01111100	8-bit displ			7 + m or 3	7 + m or 3		r
Full Displacement	00001111	10001100	full displacement		7 + m or 3	7 + m or 3		r
JNL/JGE = Jump on Not Less/Greater or Equal								
8-Bit Displacement	01111101	8-bit displ			7 + m or 3	7 + m or 3		r
Full Displacement	00001111	10001101	full displacement		7 + m or 3	7 + m or 3		r
JLE/JNG = Jump on Less or Equal/Not Greater								
8-Bit Displacement	01111110	8-bit displ			7 + m or 3	7 + m or 3		r
Full Displacement	00001111	10001110	full displacement		7 + m or 3	7 + m or 3		r
JNLE/JG = Jump on Not Less or Equal/Greater								
8-Bit Displacement	01111111	8-bit displ			7 + m or 3	7 + m or 3		r
Full Displacement	00001111	10001111	full displacement		7 + m or 3	7 + m or 3		r
JCXZ = Jump on CX Zero	11100011	8-bit displ			9 + m or 5	9 + m or 5		r
JECXZ = Jump on ECX Zero	11100011	8-bit displ			9 + m or 5	9 + m or 5		r
(Address Size Prefix Differentiates JCXZ from JECXZ)								
LOOP = Loop CX Times	11100010	8-bit displ			11 + m	11 + m		r
LOOPZ/LOOPE = Loop with Zero/Equal	11100001	8-bit displ			11 + m	11 + m		r
LOOPNZ/LOOPNE = Loop While Not Zero	11100000	8-bit displ			11 + m	11 + m		r
CONDITIONAL BYTE SET								
NOTE: Times Are Register/Memory								
SETO = Set Byte on Overflow								
To Register/Memory	00001111	10010000	mod 000	r/m	4/5	4/5		h
SETNO = Set Byte on Not Overflow								
To Register/Memory	00001111	10010001	mod 000	r/m	4/5	4/5		h
SETB/SETNAE = Set Byte on Below/Not Above or Equal								
To Register/Memory	00001111	10010010	mod 000	r/m	4/5	4/5		h

Tabel 7/4.1-23i: Samenvatting van de 386DX instructieset, deel 9.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT			CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
CONDITIONAL BYTE SET (Continued)							
SETNB = Set Byte on Not Below/Above or Equal							
To Register/Memory	00001111	10010011	mod 000 r/m	4/5	4/5		h
SETE/SETZ = Set Byte on Equal/Zero							
To Register/Memory	00001111	10010100	mod 000 r/m	4/5	4/5		h
SETNE/SETNZ = Set Byte on Not Equal/Not Zero							
To Register/Memory	00001111	10010101	mod 000 r/m	4/5	4/5		h
SETBE/SETNA = Set Byte on Below or Equal/Not Above							
To Register/Memory	00001111	10010110	mod 000 r/m	4/5	4/5		h
SETNBE/SETA = Set Byte on Not Below or Equal/Above							
To Register/Memory	00001111	10010111	mod 000 r/m	4/5	4/5		h
SETS = Set Byte on Sign							
To Register/Memory	00001111	10011000	mod 000 r/m	4/5	4/5		h
SETNS = Set Byte on Not Sign							
To Register/Memory	00001111	10011001	mod 000 r/m	4/5	4/5		h
SETP/SETPE = Set Byte on Parity/Parity Even							
To Register/Memory	00001111	10011010	mod 000 r/m	4/5	4/5		h
SETNP/SETPO = Set Byte on Not Parity/Parity Odd							
To Register/Memory	00001111	10011011	mod 000 r/m	4/5	4/5		h
SETL/SETNGE = Set Byte on Less/Not Greater or Equal							
To Register/Memory	00001111	10011100	mod 000 r/m	4/5	4/5		h
SETNL/SETGE = Set Byte on Not Less/Greater or Equal							
To Register/Memory	00001111	01111101	mod 000 r/m	4/5	4/5		h
SETLE/SETNG = Set Byte on Less or Equal/Not Greater							
To Register/Memory	00001111	10011110	mod 000 r/m	4/5	4/5		h
SETNLE/SETG = Set Byte on Not Less or Equal/Greater							
To Register/Memory	00001111	10011111	mod 000 r/m	4/5	4/5		h
ENTER = Enter Procedure	11001000	16-bit displacement, 8-bit level					
L = 0				10	10	b	h
L = 1				12	12	b	h
L > 1				15 + 4(n − 1)	15 + 4(n − 1)	b	h
LEAVE = Leave Procedure	11001001			4	4	b	h

Tabel 7/4.1-23j: Samenvatting van de 386DX instructieset, deel 10.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS					
INT = Interrupt:					
Type Specified	11001101 | type	37		b	
Type 3	11001100	33		b	
INTO = Interrupt 4 If Overflow Flag Set	11001110				
If OF = 1		35		b, e	
If OF = 0		3	3	b, e	
Bound = Interrupt 5 If Detect Value Out of Range	01100010 | mod reg r/m				
If Out of Range		44		b, e	e, g, h, j, k, r
If In Range		10	10	b, e	e, g, h, j, k, r
Protected Mode Only (INT)					
INT: Type Specified					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			282		g, j, k, r
From 80286 Task to 386DX™ CPU TSS via Task Gate			309		g, j, k, r
From 80286 Task to virt 8086 md via Task Gate			226		g, j, k, r
From 386DX™ CPU Task to 80286 TSS via Task Gate			284		g, j, k, r
From 386DX™ CPU Task to 386DX™ CPU TSS via Task Gate			311		g, j, k, r
From 386DX™ CPU Task to virt 8086 md via Task Gate			228		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			289		g, j, k, r
From virt 8086 md to 386DX™ CPU TSS via Task Gate			316		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INT: TYPE 3					
Via Interrupt or Trap Gate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			278		g, j, k, r
From 80286 Task to 386DX™ CPU TSS via Task Gate			305		g, j, k, r
From 80286 Task to Virt 8086 md via Task Gate			222		g, j, k, r
From 386DX™ CPU Task to 80286 TSS via Task Gate			280		g, j, k, r
From 386DX™ CPU Task to 386DX™ CPU TSS via Task Gate			307		g, j, k, r
From 386DX™ CPU Task to Virt 8086 md via Task Gate			224		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			285		g, j, k, r
From virt 8086 md to 386DX™ CPU TSS via Task Gate			312		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INTO:					
Via Interrupt or Trap Grate to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			280		g, j, k, r
From 80286 Task to 386DX™ CPU TSS via Task Gate			307		g, j, k, r
From 80286 Task to virt 8086 md via Task Gate			224		g, j, k, r
From 386DX™ CPU Task to 80286 TSS via Task Gate			282		g, j, k, r
From 386DX™ CPU Task to 386DX™ CPU TSS via Task Gate			309		g, j, k, r
From 386DX™ CPU Task to virt 8086 md via Task Gate			225		g, j, k, r
From virt 8086 md to 80286 TSS via Task Gate			287		g, j, k, r
From virt 8086 md to 386DX™ CPU TSS via Task Gate			314		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		

Tabel 7/4.1-23k: Samenvatting van de 386DX instructieset, deel 11.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT		NOTES	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
INTERRUPT INSTRUCTIONS *(Continued)*					
BOUND:					
Via Interrupt or Trap Gate					
to Same Privilege Level			59		g, j, k, r
Via Interrupt or Trap Gate					
to Different Privilege Level			99		g, j, k, r
From 80286 Task to 80286 TSS via Task Gate			254		g, j, k, r
From 80286 Task to 386DX™ CPU TSS via Task Gate			284		g, j, k, r
From 80268 Task to virt 8086 Mode via Task Gate			231		g, j, k, r
From 386DX™ CPU Task to 80286 TSS via Task Gate			264		g, j, k, r
From 386DX™ CPU Task to 386DX™ CPU TSS via Task Gate			294		g, j, k, r
From 80368 Task to virt 8086 Mode via Task Gate			243		g, j, k, r,
From virt 8086 Mode to 80286 TSS via Task Gate			264		g, j, k, r
From virt 8086 Mode to 386DX™ CPU TSS via Task Gate			294		g, j, k, r
From virt 8086 md to priv level 0 via Trap Gate or Interrupt Gate			119		
INTERRUPT RETURN					
IRET = Interrupt Return	11001111	22			g, h, j, k, r
Protected Mode Only (IRET)					
To the Same Privilege Level (within task)			38		g, h, j, k, r
To Different Privilege Level (within task)			82		g, h, j, k, r
From 80286 Task to 80286 TSS			232		h, j, k, r
From 80286 Task to 386DX™ CPU TSS			265		h, j, k, r
From 80286 Task to Virtual 8086 Task			213		h, j, k, r
From 80286 Task to Virtual 8086 Mode (within task)			60		
From 386DX™ CPU Task to 80286 TSS			271		h, j, k, r
From 386DX™ CPU Task to 386DX™ CPU TSS			275		h, j, k, r
From 386DX™ CPU Task to Virtual 8086 Task			223		h, j, k, r
From 386DX™ CPU Task to Virtual 8086 Mode (within task)			60		
PROCESSOR CONTROL					
HLT = HALT	11110100	5	5		l
MOV = Move to and From Control/Debug/Test Registers					
CR0/CR2/CR3 from register	00001111 00100010 11 eee reg	11/4/5	11/4/5		l
Register From CR0-3	00001111 00100000 11 eee reg	6	6		l
DR0-3 From Register	00001111 00100011 11 eee reg	22	22		l
DR6-7 From Register	00001111 00100011 11 eee reg	16	16		l
Register from DR6-7	00001111 00100001 11 eee reg	14	14		l
Register from DR0-3	00001111 00100001 11 eee reg	22	22		l
TR6-7 from Register	00001111 00100110 11 eee reg	12	12		l
Register from TR6-7	00001111 00100100 11 eee reg	12	12		l
NOP = No Operation	10010000	3	3		
WAIT = Wait until BUSY# pin is negated	10011011	7	7		

Tabel 7/4.1-23l: Samenvatting van de 386DX instructieset, deel 12.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION	FORMAT	CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
PROCESSOR EXTENSION INSTRUCTIONS					
Processor Extension Escape	11011TTT / mod LLL r/m TTT and LLL bits are opcode information for coprocessor.	See 80287/80387 data sheets for clock counts			h
PREFIX BYTES					
Address Size Prefix	01100111	0	0		
LOCK = Bus Lock Prefix	11110000	0	0		m
Operand Size Prefix	01100110	0	0		
Segment Override Prefix					
CS:	00101110	0	0		
DS:	00111110	0	0		
ES:	00100110	0	0		
FS:	01100100	0	0		
GS:	01100101	0	0		
SS:	00110110	0	0		
PROTECTION CONTROL					
ARPL = Adjust Requested Privilege Level					
From Register/Memory	01100011 / mod reg r/m	N/A	20/21	a	h
LAR = Load Access Rights					
From Register/Memory	00001111 / 00000010 / mod reg r/m	N/A	15/16	a	g, h, j, p
LGDT = Load Global Descriptor					
Table Register	00001111 / 00000001 / mod 010 r/m	11	11	b, c	h, l
LIDT = Load Interrupt Descriptor					
Table Register	00001111 / 00000001 / mod 011 r/m	11	11	b, c	h, l
LLDT = Load Local Descriptor					
Table Register to Register/Memory	00001111 / 00000000 / mod 010 r/m	N/A	20/24	a	g, h, j, l
LMSW = Load Machine Status Word					
From Register/Memory	00001111 / 00000001 / mod 110 r/m	11/14	11/14	b, c	h, l
LSL = Load Segment Limit					
From Register/Memory	00001111 / 00000011 / mod reg r/m				
Byte-Granular Limit		N/A	21/22	a	g, h, j, p
Page-Granular Limit		N/A	25/26	a	g, h, j, p
LTR = Load Task Register					
From Register/Memory	00001111 / 00000000 / mod 001 r/m	N/A	23/27	a	g, h, j, l
SGDT = Store Global Descriptor					
Table Register	00001111 / 00000001 / mod 000 r/m	9	9	b, c	h
SIDT = Store Interrupt Descriptor					
Table Register	00001111 / 00000001 / mod 001 r/m	9	9	b, c	h
SLDT = Store Local Descriptor Table Register					
To Register/Memory	00001111 / 00000000 / mod 000 r/m	N/A	2/2	a	h

Tabel 7/4.1-23m: Samenvatting van de 386DX instructieset, deel 13.

4.1 80386

386DX™ Microprocessor Instruction Set Clock Count Summary (Continued)

INSTRUCTION		FORMAT			CLOCK COUNT: Real Address Mode or Virtual 8086 Mode	CLOCK COUNT: Protected Virtual Address Mode	NOTES: Real Address Mode or Virtual 8086 Mode	NOTES: Protected Virtual Address Mode
SMSW	= Store Machine Status Word	0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 1	mod 1 0 0 r/m	2/2	2/2	b, c	h, l
STR	= Store Task Register							
	To Register/Memory	0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	mod 0 0 1 r/m	N/A	2/2	a	h
VERR	= Verify Read Access							
	Register/Memory	0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	mod 1 0 0 r/m	N/A	10/11	a	g, h, j, p
VERW	Verify Write Access	0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	mod 1 0 1 r/m	N/A	15/16	a	g, h, j, p

INSTRUCTION NOTES FOR TABLE

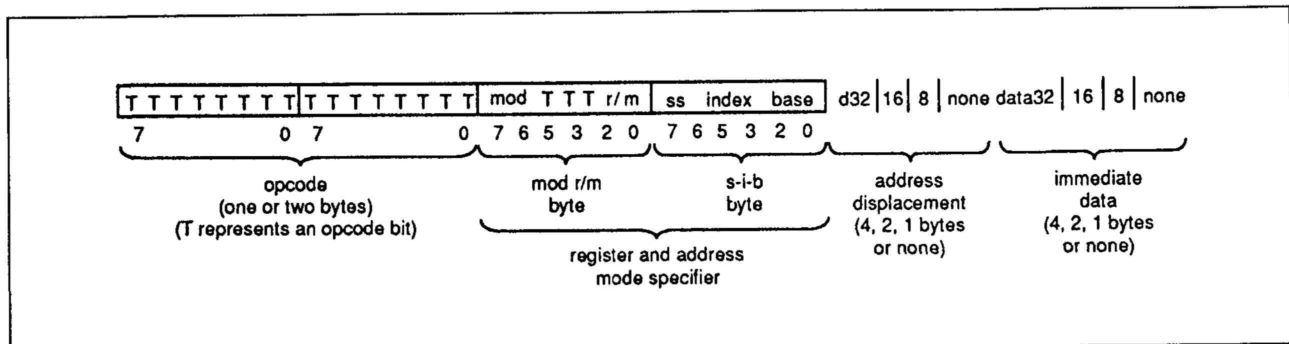
Notes a through c apply to 386DX Microprocessor Real Address Mode only:
a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid opcode).
b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS or GS limit, FFFFH. Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.
c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Notes d through g apply to 386DX Microprocessor Real Address Mode and 386DX Microprocessor Protected Virtual Address Mode:
d. The 386DX Microprocessor uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).
Clock counts given are minimum to maximum. To calculate actual clocks use the following formula:
Actual Clock = if $m <> 0$ then max ($[\log_2 |m|]$, 3) + b clocks:
if $m = 0$ then $3 + b$ clocks
In this formula, m is the multiplier, and
b = 9 for register to register,
b = 12 for memory to register,
b = 10 for register with immediate to register,
b = 11 for memory with immediate to register.
e. An exception may occur, depending on the value of the operand.
f. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK# prefix.
g. LOCK# is asserted during descriptor table accesses.

Notes h through r apply to 386DX Microprocessor Protected Virtual Address Mode only:
h. Exception 13 fault (general protection violation) will occur if the memory operand in CS, DS, ES, FS or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, an exception 12 (stack segment limit violation or not present) occurs.
i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present) occurs.
j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.
k. JMP, CALL, INT, RET and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is violated.
l. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).
m. An exception 13 fault occurs if CPL is greater than IOPL.
n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.
o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 if desiring to reset the PE bit.
p. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather, the zero flag is cleared.
q. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or not present) will occur if the stack limit is violated by the operand's starting address.
r. The destination of a JMP, CALL, INT, RET or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.

Tabel 7/4.1-23n: Samenvatting van de 386DX instructieset, deel 14.

4.1 80386



Figuur 7/4.1-42: Algemeen instructie-formaat.

Instructie-codering

Alle instructie-coderingen zijn subsets van het algemene instructie-formaat in figuur 7/4.1-42. Instructies bestaan uit één of twee primaire opcode-bytes, eventueel een adres-specificeerder (die bestaat uit het "mod r/m" byte en het "scaled index" byte), een displacement (als die nodig is) en een immediate data-field (indien nodig).

Binnen de primaire opcode(s) kunnen kleinere codeer-velden worden gedefinieerd. Deze velden variëren naar gelang de klasse van operatie. Ze stellen informatie vast zoals richting van de operatie, afmetingen van de displacements, register-codering of teken-uitbreiding. Bijna alle instructies die naar een operand in geheugen verwijzen hebben een adresseer-mode byte, volgend na de primaire opcode-byte(s). Dit byte (het mod r/m byte) specificeert de adresseer-mode die gebruikt moet worden. Bepaalde coderingen van het mod r/m byte geven een tweede adresseer-byte (het scaled-index-base byte) om de adresseer-mode volledig te specificeren.

Adresseer-modes kunnen een displacement bevatten die onmiddellijk na het mod r/m byte of het scaled-index byte komt. Als een displacement aanwezig is, zijn 8, 16 of 32 bits mogelijk. Als de instructie een immediate operand specificeert, komt de immediate operand na eventuele displacement-bytes. Als een immediate operand is gespecificeerd, is deze altijd het laatste veld van de instructie. In figuur 7/4.1-42 zijn verschillende velden te zien die in een instructie kunnen voorkomen, maar niet alle velden. In sommige instructies komen ook enkele kleinere velden voor, soms binnen de opcode-bytes zelf. Tabel 7/4.1-24 geeft een complete lijst met alle velden die in de 386DX instructieset voorkomen.

32 bit uitbreidingen van de instructieset

Bij de 386DX microprocessor is de 8086/80186/80286 instructieset uitgebreid in twee orthogonale richtingen: voor het ondersteunen van de 32 bit datatypen zijn 32 bit vormen van alle 16 bit instructies toegevoegd en er zijn 32 bit adresseringsmodes beschikbaar voor alle instructies die op geheugen betrekking hebben. Deze orthogonale uitbreiding van de instructieset werd bereikt door een "default" (D) bit in de code-segment descriptor op te nemen en twee prefixes aan de instructieset toe te voegen.

Of de instructie met 16 of 32 bits werkt hangt af van het D-bit in de code-segment descriptor die de default-lengte van zowel operands als effectieve adressen bepaalt. In de Real Address Mode of Virtual 8086 Mode worden geen code-segment descriptors gebruikt, maar wordt door de 386DX processor een D-waarde van 0 verondersteld (voor 16 bit default-lengten die overeenkomen met de 8086/80186/80286). Twee prefixes, de Operand Size Prefix en de Effective Address Size Prefix, maken individuele overschrijving van de Default-selectie mogelijk. Deze prefixes (één of beide) mogen vóór elke opcode byte worden geplaatst en beïnvloeden alleen de desbetreffende instructie.

4.1 80386

Field Name	Description	Number of Bits
w	Specifies if Data is Byte or Full Size (Full Size is either 16 or 32 bits)	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod; 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Condition Instructions, specifies a Condition Asserted or a Condition Negated	4

Tabel 7/4.1-24: Velden binnen de 386DX instructies.

De operand-lengte of de effectieve adreslengte wordt door deze operands omgeschakeld naar de "tegenoverliggende" waarde van de default-waarde. Als de operand default-waarde bijvoorbeeld is ingesteld voor 32 bit operaties, maakt een aanwezige Operand Size Prefix dat de instructie wordt overgeschakeld naar een 16 bit data-operatie. Of, als de default effectieve adreslengte 16 bit is, laat een Effective Address Size Prefix de instructie gebruik maken van 32 bit effectieve adres-berekeningen.
Deze 32 bit uitbreidingen zijn beschikbaar in alle 386DX modes, inclusief de Real Address Mode of de Virtual 8086 Mode. In deze modes is de default-waarde altijd 16 bit, zodat prefixes nodig zijn om 32 bit operands en adressen te specificeren.
Voor instructies met meer dan één prefix is de volgorde van de prefixes niet belangrijk.

w Field	Operand Size During 16-Bit Data Operations	Operand Size During 32-Bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

Tabel 7/4.1-25: Bepaling van de operand-lengte door middel van het w-veld.

Instructies met 8 bit en 16 bit operands beïnvloeden de inhoud van de hogere bits van de uitgebreide registers over het algemeen niet.

Codering van instructievelden
Binnen de instructie bevinden zich verschillende velden die registerkeuze, adresseermode, enzovoorts aangeven. De precieze codering van deze velden volgt hierna.

Codering van het operand-lengte (w) veld
Elke instructie die een data-operatie uitvoert, werkt als een 32 bit of een 16 bit operatie. Binnen de beperkingen van de operatielengte codeert het w-veld de operand-lengte als één byte of als de volledige operatie lengte (zie tabel 7/4.1-25).

Codering van het algemeen register-veld
Het algemeen register (General Register) wordt gespecificeerd met het registerveld dat kan voorkomen in de primaire opcode bytes, als het registerveld van de "mod r/m" byte, als het r/m-veld of als de "mod r/m" byte.

Codering van het segment-register (sreg) veld
Het segment-register-veld (sreg field) in sommige instructies is een 2 bit veld waar-

4.1 80386

mee één van de vier 80286 segment-registers kan worden gespecificeerd. Het sreg field in andere instructies is een 3 bit veld, voor specificatie van de 386DX FS en GS segment-registers.

reg Field	Register Selected During 16-Bit Data Operations	Register Selected During 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI

Tabel 7/4.1-26: Codering van het registerveld wanneer het w-veld niet aanwezig is in de instructie.

Register Specified by reg Field During 16-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Tabel 7/4.1-27: Codering van het registerveld wanneer het w-veld wel aanwezig is in de instructie.

Codering van de adresserings-mode

Behalve voor speciale instructies, zoals PUSH en POP waarbij de adresseringsmode al vast staat, wordt de adresseer-mode voor de lopende instructie bepaald door de adresseer-bytes die na de primaire opcode zijn opgenomen. Het primaire adresseer-byte is het "mod r/m" byte, terwijl tevens een tweede byte met adresserings-informatie, het "s-i-b" (scaled index base) byte kan worden gespecificeerd.

Het sib-byte wordt gespecificeerd wanneer de 32 bit adresseringsmode wordt gebruikt en het "mod r/m" byte r/m = 100 en mod = 00, 01 of 10 heeft. Als het sib-byte aanwezig is, is de 32 bit adresseringsmode een functie van de mod-, ss-, index- en basis-velden.

Het primaire adresserings-byte, het "mod r/m" byte, bevat ook drie bits (TTT in figuur 7/4.1-42) die soms worden gebruikt als uitbreiding op de primaire opcode. De drie bits mogen echter ook worden gebruikt als een registerveld. In de volgende tabellen zijn de coderingen voor de berekening van alle 16 en 32 bit adressen te zien.

Register Specified by reg Field During 32-Bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

Tabel 7/4.1-28: Specificatie van het registerveld tijdens 32 bit data-operaties.

2-Bit sreg2 Field

2-Bit sreg2 Field	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

Tabel 7/4.1-29: Selectie met het 2 bit sreg2 veld.

4.1 80386

3-Bit sreg3 Field

3-Bit sreg3 Field	Segment Register Selected
000	ES
001	CS
010	SS
011	DS
100	FS
101	GS
110	do not use
111	do not use

Tabel 7/4.1-30: Selectie met het 3 bit sreg3 veld.

mod r/m	Effective Address
00 000	DS:[BX + SI]
00 001	DS:[BX + DI]
00 010	SS:[BP + SI]
00 011	DS:[BP + DI]
00 100	DS:[SI]
00 101	DS:[DI]
00 110	DS:d16
00 111	DS:[BX]
01 000	DS:[BX + SI + d8]
01 001	DS:[BX + DI + d8]
01 010	SS:[BP + SI + d8]
01 011	SS:[BP + DI + d8]
01 100	DS:[SI + d8]
01 101	DS:[DI + d8]
01 110	SS:[BP + d8]
01 111	DS:[BX + d8]
10 000	DS:[BX + SI + d16]
10 001	DS:[BX + DI + d16]
10 010	SS:[BP + SI + d16]
10 011	SS:[BP + DI + d16]
10 100	DS:[SI + d16]
10 101	DS:[DI + d16]
10 110	SS:[BP + d16]
10 111	DS:[BX + d16]
11 000	Register—See Below
11 001	Register—See Below
11 010	Register—See Below
11 011	Register—See Below
11 100	Register—See Below
11 101	Register—See Below
11 110	Register—See Below
11 111	Register—See Below

Tabel 7/4.1-31: Codering van de 16 bit adresseringsmode met het "mod r/m" byte.

Register Specified by r/m During 16-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-Bit Data Operations

mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Tabel 7/4.1-32: Boven: specificatie van de registers door r/m bij 16 bit data-operaties.
Onder: specificatie van de registers door r/m bij 32 bit data-operaties.

4.1 80386

mod r/m	Effective Address
00 000	DS:[EAX]
00 001	DS:[ECX]
00 010	DS:[EDX]
00 011	DS:[EBX]
00 100	s-i-b is present
00 101	DS:d32
00 110	DS:[ESI]
00 111	DS:[EDI]
01 000	DS:[EAX + d8]
01 001	DS:[ECX + d8]
01 010	DS:[EDX + d8]
01 011	DS:[EBX + d8]
01 100	s-i-b is present
01 101	SS:[EBP + d8]
01 110	DS:[ESI + d8]
01 111	DS:[EDI + d8]
10 000	DS:[EAX + d32]
10 001	DS:[ECX + d32]
10 010	DS:[EDX + d32]
10 011	DS:[EBX + d32]
10 100	s-i-b is present
10 101	SS:[EBP + d32]
10 110	DS:[ESI + d32]
10 111	DS:[EDI + d32]
11 000	Register—See Below
11 001	Register—See Below
11 010	Register—See Below
11 011	Register—See Below
11 100	Register—See Below
11 101	Register—See Below
11 110	Register—See Below
11 111	Register—See Below

Tabel 7/4.1-33: Codering van de 32 bit adresseringsmode met het "mod r/m" byte (geen "s-i-b" byte aanwezig).

Register Specified by reg or r/m During 16-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by reg or r/m During 32-Bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI

Tabel 7/4.1-34: Boven: specificatie van de registers door reg of r/m tijdens 16 bit data-operaties.
Onder: specificatie van de registers door reg of r/m tijdens 32 bit data-operaties.

mod base	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

Note: Mod field in mod r/m byte; ss, index, base fields in s-i-b byte.

Tabel 7/4.1-35: Codering van de 32 bit adresseringsmode ("mod r/m" byte en "s-i-b" byte aanwezig).

Codering van het operatie-richtings (d) veld

Bij vele twee-operand instructies is het d-veld aanwezig om aan te geven welke operand als bron wordt beschouwd en welke als doel (tabel 7/4.1-37).

Codering van het teken-uitbreidings (s) veld

Het s-veld komt hoofdzakelijk voor bij instructies met immediate data-velden. Het s-veld heeft alleen invloed als de lengte van de immediate data 8 bits is en in een 16 bit of 32 bit bestemming wordt geplaatst.

Codering van het voorwaardelijke test (tttn) veld

Voor de voorwaardelijke instructies (conditional jumps en set-on-condition) wordt tttn gebruikt. Hierbij geeft n aan of de conditie (n = 0) wordt gebruikt of de negatie (n = 1), terwijl ttt de te testen conditie geeft (zie tabel 7/4.1-39).

Codering van het Control-, Debug- of Test-register (eee) veld

Dit veld wordt gebruikt voor het laden en opslaan van de besturings-, debug- en testregisters.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

Index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg (see note)
101	EBP
110	ESI
111	EDI

Note: When index field is 100, indicating no index register, then ss field must equal 00. If index is 100 and ss does not equal 00, the effective address is undefined.

Tabel 7/4.1-36: Boven: schaal-factor als functie van ss.
Onder: index-register als functie van index.

d	Direction of Operation
0	Register/Memory ← Register reg Field indicates Source Operand; mod r/m or mod ss index base indicates Destination Operand.
1	Register ← Register Memory reg Field indicates Destination Operand; mod r/m or mod ss index base indicates Source Operand.

Tabel 7/4.1-37: Bepaling van de richting met het d-veld.

4.1 80386

s	Effect on Immediate Data 8	Effect on Immediate Data 16|32
0	None	None
1	Sign-Extended Data 8 to fill 16-Bit or 32-Bit Destination	None

Tabel 7/4.1-38: Codering van de teken-uitbreiding met het s-veld.

Mnemonic	Condition	tttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Not Greater Than	1110
NLE/G	Not Less Than or Equal/Greater Than	1111

Tabel 7/4.1-39: Testbare condities.

When Interpreted as Control Register Field

eee Code	Reg Name
000	CR0
010	CR2
011	CR3
Do not use any other encoding.	

When Interpreted as Debug Register Field

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7
Do not use any other encoding.	

When Interpreted as Test Register Field

eee Code	Reg Name
110	TR6
111	TR7
Do not use any other encoding.	

Tabel 7/4.1-40: Van boven naar onder:
bij interpretatie als Control Register-veld;
bij interpretatie als Debug Register-veld;
bij interpretatie als Test Register-veld.

Overige kenmerken

Voeding

Hoewel de 386DX in CMOS is uitgevoerd kunnen, als gevolg van de hoge clock-frequentie en de 72 uitgangsbuffers, schommelingen op de voedingslijn ontstaan. Om de verdeling van de voeding over de microprocessor zo geleidelijk mogelijk te maken is de chip uitgevoerd met 20 V_{CC} en 21 V_{SS} pennen die allemaal aangesloten dienen te worden. De printplaat waarop de microprocessor wordt gemonteerd moet bovendien zijn voorzien van een V_{CC}- en een GND-vlak. De ontkoppeling van de voeding moet zo dicht mogelijk bij de processor plaatsvinden.

Optrekweerstanden

De $\overline{\text{ERROR}}$- en $\overline{\text{BUSY}}$-ingangen hebben interne optrekweerstanden van ongeveer 20 kΩ om te voorkomen dat deze ingangen meedoen wanneer geen 387DX coprocessor aanwezig is.
De $\overline{\text{BS16}}$-ingang heeft ook een interne optrekweerstand van 20 kΩ en de $\overline{\text{PEREQ}}$-ingang wordt intern neergetrokken met 20 kΩ.

Niet gebruikte pennen

Voor een betrouwbare werking moeten alle niet gebruikte ingangen altijd met een geschikt signaal-niveau worden verbonden (maar N.C.-pennen dienen los te blijven!).

4.1 80386

Elektrische eigenschappen en timing-karakteristieken

Tenslotte zijn in de tabellen 7/4.1-41, -42 en -43 en de bijbehorende figuren 7/4.1-43 tot en met -48 de elektrische eigenschappen en de timing-karakteristieken van de 386DX processor (33 MHz-type) opgenomen.

Parameter	386™ DX 20, 25, 33 MHz Maximum Rating
Storage Temperature	−65°C to +150°C
Case Temperature Under Bias	−65°C to +110°C
Supply Voltage with Respect to V_{SS}	−0.5V to +6.5V
Voltage on Other Pins	−0.5V to V_{CC} + 0.5V

Tabel 7/4.1-41: Maximaal toelaatbare waarden.

Symbol	Parameter	386™ DX 20 MHz, 25 MHz, 33 MHz		Unit	Test Conditions
		Min	Max		
V_{IL}	Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{ILC}	CLK2 Input Low Voltage	−0.3	0.8	V	(Note 1)
V_{IHC}	CLK2 Input High Voltage 20 MHz 25 MHz and 33 MHz	 V_{CC} − 0.8 3.7	 V_{CC} + 0.3 V_{CC} + 0.3	 V V	
V_{OL}	Output Low Voltage I_{OL} = 4 mA: A2–A31, D0–D31 I_{OL} = 5 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA		 0.45 0.45	 V V	
V_{OH}	Output High Voltage I_{OH} = 1 mA: A2–A31, D0–D31 I_{OH} = 0.9 mA: BE0#–BE3#, W/R#, D/C#, M/IO#, LOCK#, ADS#, HLDA	 2.4 2.4		 V V	
I_{LI}	Input Leakage Current (For All Pins except BS16#, PEREQ, BUSY#, and ERROR#)		±15	μA	0V ≤ V_{IN} ≤ V_{CC}
I_{IH}	Input Leakage Current (PEREQ Pin)		200	μA	V_{IH} = 2.4V (Note 2)
I_{IL}	Input Leakage Current (BS16#, BUSY#, and ERROR# Pins)		−400	μA	V_{IL} = 0.45 (Note 3)
I_{LO}	Output Leakage Current		±15	μA	0.45V ≤ V_{OUT} ≤ V_{CC}
I_{CC}	Supply Current CLK2 = 40 MHz: with 20 MHz 386™ DX CLK2 = 50 MHz: with 25 MHz 386™ DX CLK2 = 66 MHz: with 33 MHz 386™ DX		 500 550 550	 mA mA mA	 I_{CC} Typ. = 460 mA I_{CC} Typ. = 500 mA I_{CC} Typ. = 400 mA
C_{IN}	Input or I/O Capacitance		10	pF	F_C = 1 MHz (Note 4)
C_{OUT}	Output Capacitance		12	pF	F_C = 1 MHz (Note 4)
C_{CLK}	CLK2 Capacitance		20	pF	F_C = 1 MHz (Note 4)

NOTES:
1. The min value, −0.3, is not 100% tested.
2. PEREQ input has an internal pulldown resistor.
3. BS16#, BUSY# and ERROR# inputs each have an internal pullup resistor.
4. Not 100% tested.

Tabel 7/4.1-42: Gelijkspannings-kenmerken voor de 33, 25 en 20 MHz typen van de 386DX.

4.1 80386

Symbol	Parameter	33 MHz 386™ DX		Unit	Notes
		Min	Max		
	Operating Frequency	8	33.3	MHz	Half of CLK2 Frequency
t1	CLK2 Period	15.0	62.5	ns	
t2a	CLK2 High Time	6.25		ns	at 2V
t2b	CLK2 High Time	4.5		ns	at 3.7V
t3a	CLK2 Low Time	6.25		ns	at 2V
t3b	CLK2 Low Time	4.5		ns	at 0.8V
t4	CLK2 Fall Time		4	ns	3.7V to 0.8V (Note 3)
t5	CLK2 Rise Time		4	ns	0.8V to 3.7V (Note 3)
t6	A2–A31 Valid Delay	4	15	ns	C_L = 50 pF
t7	A2–A31 Float Delay	4	20	ns	(Note 1)
t8	BE0#–BE3#, LOCK# Valid Delay	4	15	ns	C_L = 50 pF
t9	BE0#–BE3#, LOCK# Float Delay	4	20	ns	(Note 1)
t10	W/R#, M/IO#, D/C#, Valid Delay	4	15	ns	C_L = 50 pF
t10a	ADS# Valid Delay	4	14.5	ns	C_L = 50 pF
t11	W/R#, M/IO#, D/C#, ADS# Float Delay	4	20	ns	(Note 1)
t12	D0–D31 Write Data Valid Delay	7	24	ns	C_L = 50 pF, (Note 4)
t12a	D0–D31 Write Data Hold Time	2			C_L = 50 pF
t13	D0–D31 Float Delay	4	17	ns	(Note 1)
t14	HLDA Valid Delay	4	20	ns	C_L = 50 pF
t15	NA# Setup Time	5		ns	
t16	NA# Hold Time	2		ns	
t17	BS16# Setup Time	5		ns	
t18	BS16# Hold Time	2		ns	
t19	READY# Setup Time	7		ns	
t20	READY# Hold Time	4		ns	

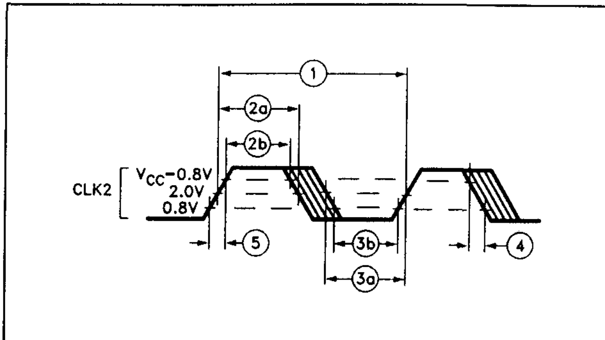
Symbol	Parameter	33 MHz 386™ DX		Unit	Notes
		Min	Max		
t21	D0–D31 Read Setup Time	5		ns	
t22	D0–D31 Read Hold Time	3		ns	
t23	HOLD Setup Time	11		ns	
t24	HOLD Hold Time	2		ns	
t25	RESET Setup Time	5		ns	
t26	RESET Hold Time	2		ns	
t27	NMI, INTR Setup Time	5		ns	(Note 2)
t28	NMI, INTR Hold Time	5		ns	(Note 2)
t29	PEREQ, ERROR#, BUSY# Setup Time	5		ns	(Note 2)
t30	PEREQ, ERROR#, BUSY# Hold Time	4		ns	(Note 2)

NOTES:

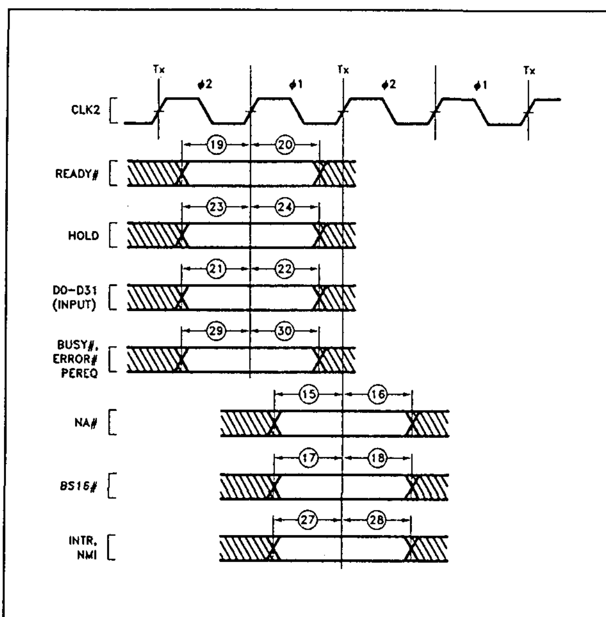
1. Float condition occurs when maximum output current becomes less than I_{LO} in magnitude. Float delay is not 100% tested.
2. These inputs are allowed to be asynchronous to CLK2. The setup and hold specifications are given for testing purposes, to assure recognition within a specific CLK2 period.
3. Rise and fall times are not tested.
4. Min. time not 100% tested.

Tabel 7/4.1-43: Schakeltijden van de 33 MHz 386DX microprocessor.

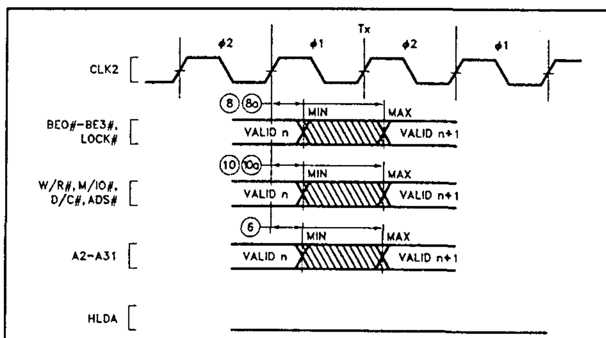
4.1 80386



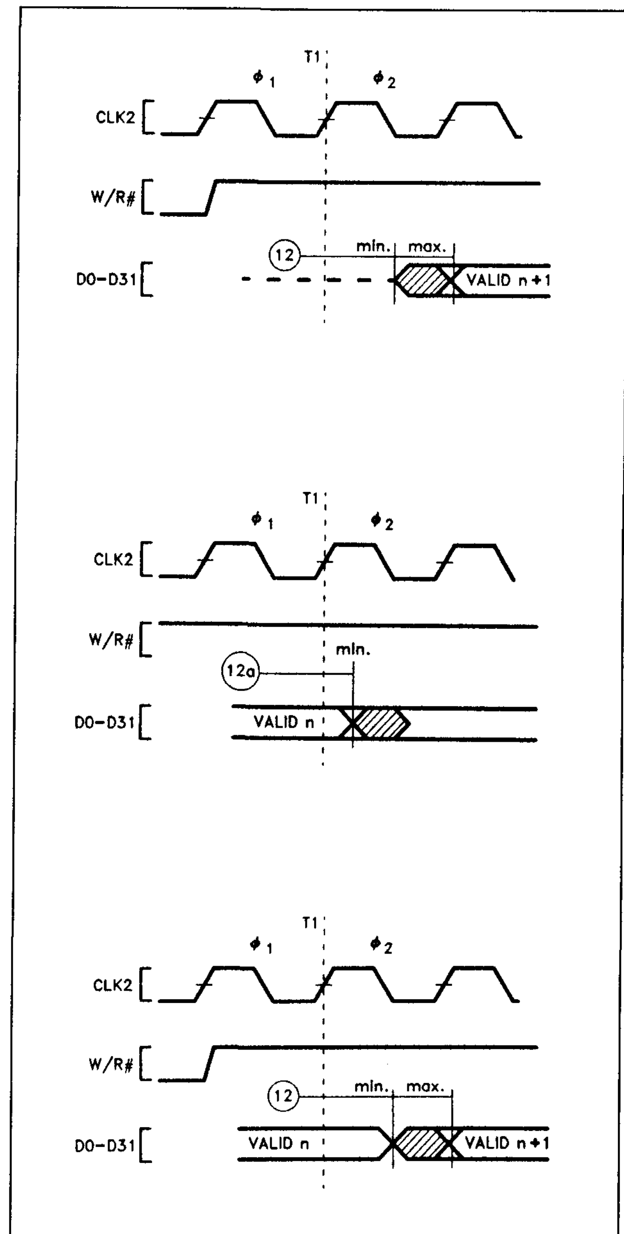
Figuur 7/4.1-43: CLK2-timing.



Figuur 7/4.1-44: Signaal-setup en -Hold timing.

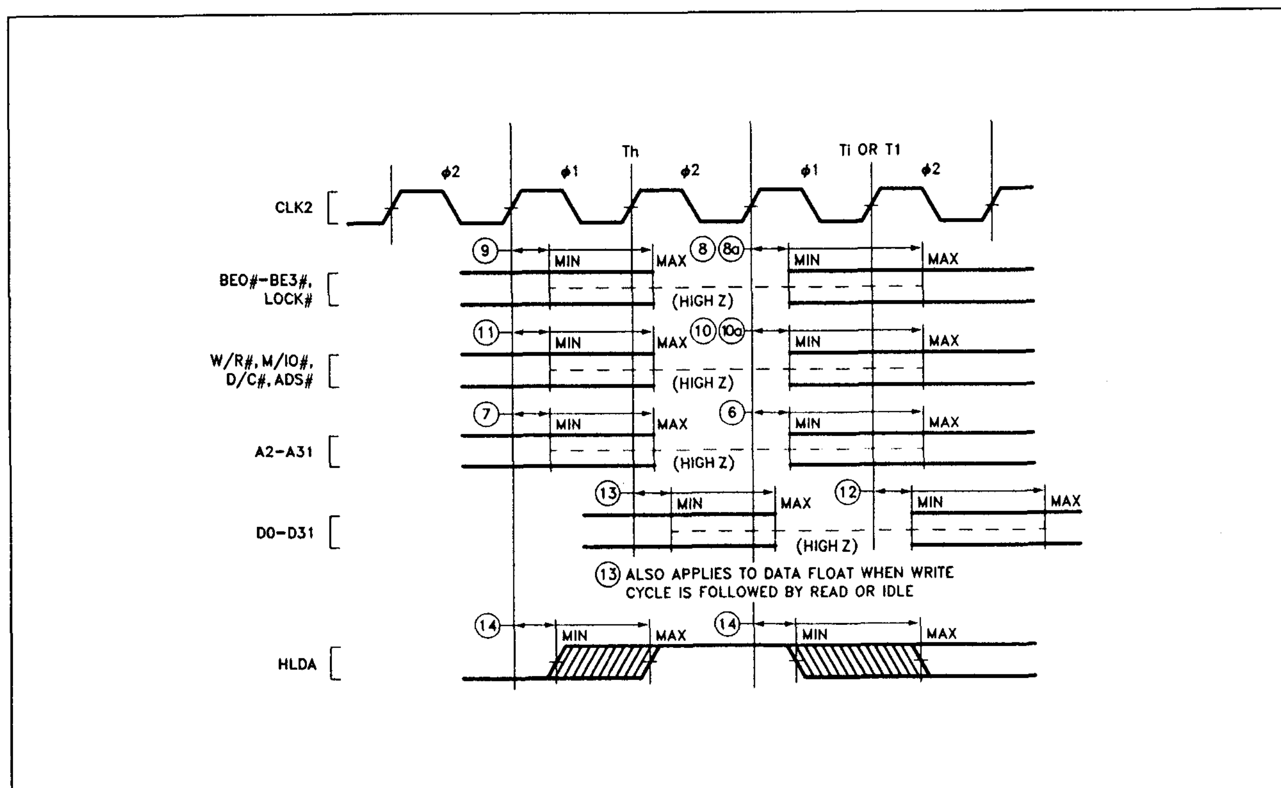


Figuur 7/4.1-45: Delay-timing voor geldige Output.

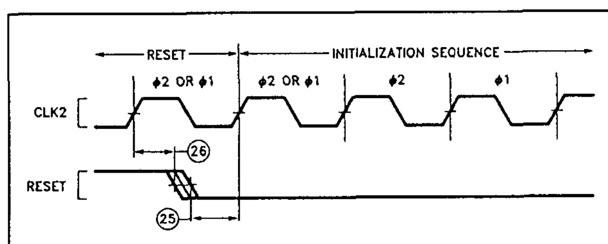


Figuur 7/4.1-46: Van boven naar onder:
Write Data Valid Delay timing (25 MHz, 33 MHz);
Write Data Hold timing (25 MHz, 33 MHz);
Write Data Valid timing (20 MHz).

4.1 80386



Figuur 7/4.1-47: Output Float Delay en HLDA Valid Delay timing.



Figuur 7/4.1-48: RESET Setup en Hold timing en interne fase.

4.1 80386

7/4.2

80486

Inleiding

Am486

De 80486 is de vierde in de reeks microprocessoren voor PC's van Intel. Aangezien de 80486 alweer is opgevolgd door de Pentium (Pro, MMX en II) hebben Intel en zijn concurrenten Cyrix en Texas Instruments de belangstelling verloren. Alleen AMD blijft nog met de verkoop doorgaan, aangezien die veel hebben geïnvesteerd in de enhanced Am486, die de snelste 486 ter wereld blijkt te zijn. In dit deel wordt dan ook de meeste aandacht besteed aan de enhanced 486 van AMD. Waar gesproken wordt van "486" betekent dit dat het onderwerp niet alleen betrekking heeft op de Am486 maar ook op de 486-typen van andere merken.

Compatibiliteit

Alle processoren in deze reeks, die met de 8088 werd begonnen, zijn gebaseerd op dezelfde architectuur, waardoor ze telkens "opwaarts compatibel" zijn: programma's die voor oudere typen zijn geschreven kunnen ook draaien op de nieuwere. Met de 80486 kunnen bedrijfssystemen als DOS, Windows, OS/2, Novell NetWare en UNIX System V/386 zeer goed presteren. De 80486 voert veel gebruikte instructies in één cyclus uit. Met behulp van speciale instructies en hardware kan hij bovendien worden toegepast in multiprocessor-systemen.

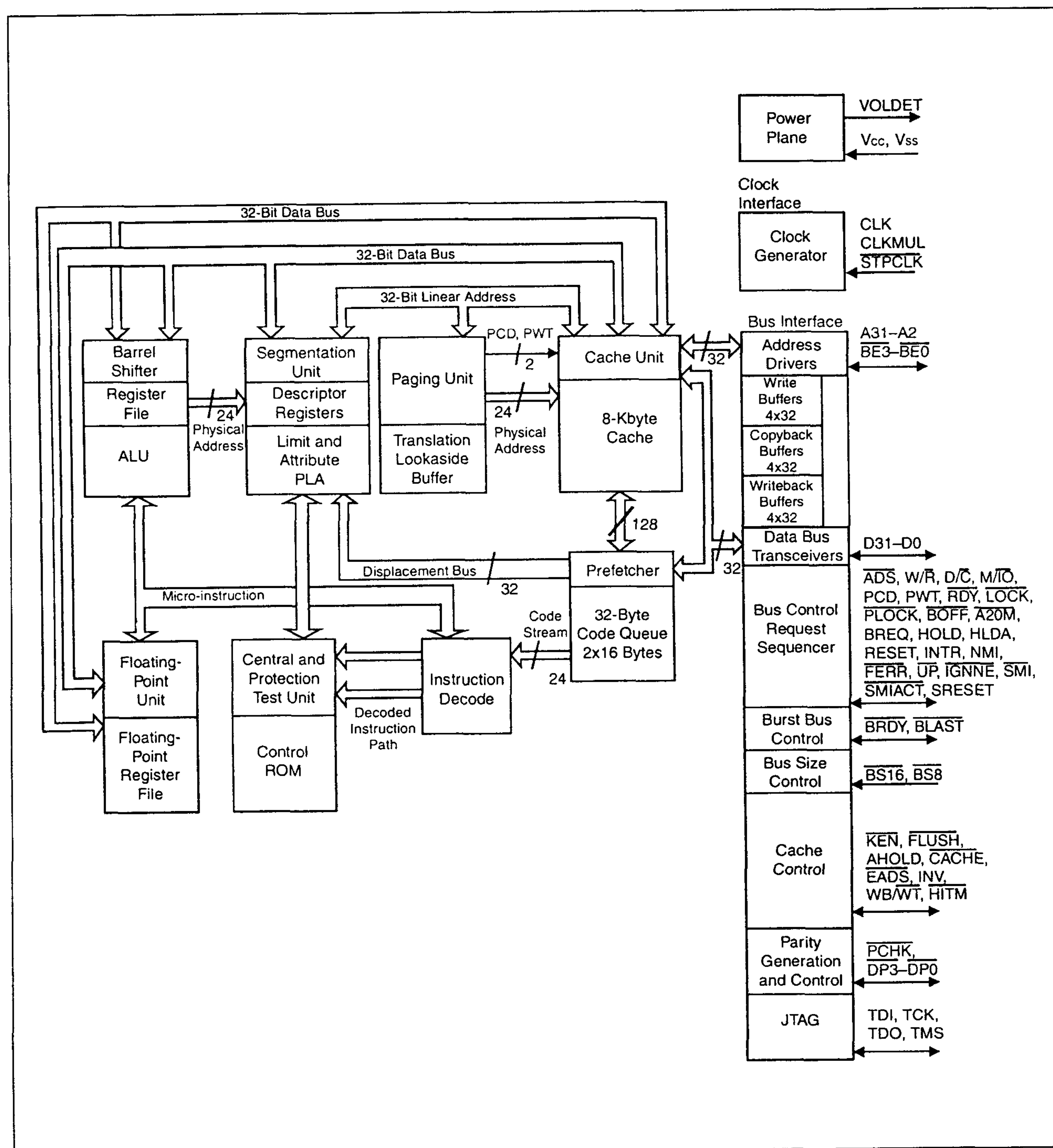
Twee versies

De i486 (80486 van Intel) was oorspronkelijk verkrijgbaar in twee uitvoeringen: de 80486DX en de 80486SX. De DX-versie is de compleetste en de chip bevat naast de logica die nodig is om compatibel te zijn met de 80386 een "Floating Point" rekeneenheid, 8 kB cache-geheugen en "Memory Management". De 80486SX is een goedkoper "instapmodel" dat geen interne rekeneenheid heeft. Beide versies hebben echter de volledige 32 bit architectuur. Wanneer veel gerekend moet worden (dus ook voor grafisch werk) wordt de DX-versie aanbevolen, omdat die een aanzienlijke tijdwinst oplevert. Overigens kan de 80486SX later nog worden uitgebreid met de 80487SX rekencoprocessor, als daarvoor tenminste een speciale (169 pens) voet op de print aanwezig is. Ook is het mogelijk de 80486SX te verwijderen en een zogenaamde "Overdrive"-processor in het voetje voor de rekenchip te steken, waardoor een snelheidsverhoging van 70 % wordt bereikt.

Snelheden

De i486 microprocessoren zijn leverbaar voor verschillende kloksnelheden. De typen 486-25, -33 en -50 werken zowel intern als extern op respectievelijk 25, 33 en 50 MHz. Daarnaast zijn er typen die net als de "Overdrive"-processoren gebruik maken van klokverdubbeling: aangezien de 486 CPU's een inwendig cache-geheugen hebben, terwijl speciale functies de externe busbandbreedte beperken, is de interne bus slechts gedurende de helft van de tijd in gebruik. De CPU's met klokverdubbelaar werken daardoor intern op een tweemaal zo hoge frequentie.

4.2 80486



Figuur 7/4.2-1: Blokschema van de enhanced Am80486 microprocessor (de i486 heeft geen JTAG-voorzieningen en ook de copy-back en write-back buffers ontbreken).

Deze processoren hebben de typeaanduiding DX2. Een 486DX2-50 processor werkt intern op 50 MHz en extern op 25 MHz (en een 486DX2-66 op 66, respectievelijk 33 MHz), waardoor bijvoorbeeld snellere (duurdere) geheugens niet nodig zijn. Overigens is ook een i486SX2-50 leverbaar en zijn er "overdrive" processoren voor zowel

4.2 80486

het SX als het DX-type beschikbaar (beide 70 % sneller). Nieuwere typen zijn de Intel-DX4 die intern op 75 MHz werkt en extern op 25 MHz en de Intel-DX400 die intern op 100 MHz werkt en extern (naar keuze) op 33 of 50 MHz (bij deze typen wordt de "naam" 80486 niet meer gebruikt). Van AMD zijn er DX2 microprocessoren die intern op 66 en 80 MHz werken en DX4-typen die intern op 75, 100 of 120 MHz werken.

Specificaties

De i486 heeft een gepijplijnde 32 bit architectuur en een tot 106 MB/s bruikbare interne burst-bus (zie ook figuur 7/4.2-1). De inwendige rekeneenheid (FPU: Floating Point Unit) implementeert de volledige IEEE 754 drijvende komma standaard en is compatibel met de 80387 mathematische coprocessor. Voor de rekenkundige opgaven is een speciale stack van acht 80 bit registers beschikbaar. De FPU werkt parallel met de integereenheid. De 486 kan 4 GB fysiek en 64 TB virtueel geheugen adresseren (net als de 386). Alle Intel 486 processoren (behalve de 486DX50) zijn "SL Enhanced": geschikt voor power management. Ze kunnen in een niet-actieve standby-mode worden gezet waardoor ze veel minder energie verbruiken.
De enhanced Am486 microprocessoren zijn rond de standaard Am486-kern gebouwd en bieden een write-back cache en verbeterd power management, inclusief SMM (System Management Mode) en clock-control. Hierdoor zijn deze 3 V typen (met 5 V tolerante I/O) ideaal voor "Energy Star" (groene) desktop en draagbare systemen. De SMM-functie wordt uitgevoerd met een standaard 2-pens interface. In de write-back mode wordt veel gebruikte data opgeborgen in de snelle interne cache en kan voortdurend worden opgevraagd totdat de data verandert.
De enhanced clock-control staat toe dat de klok onder bepaalde omstandigheden wordt gestopt om het energieverbruik nog verder te verminderen.
Bovendien kunnen de enhanced Am486 microprocessoren met de JTAG Boundary-Scan methode worden getest op het moederbord.

CPU Type	Operating Frequency	Bus Speed	Available Package
DX2	66 MHz	33 MHz	168-pin PGA
	80 MHz	40 MHz	
DX4	75 MHz	25 MHz	168-pin PGA or 208-pin SQFP
	100 MHz	33 MHz	
	120 MHz	40 MHz	168-pin PGA

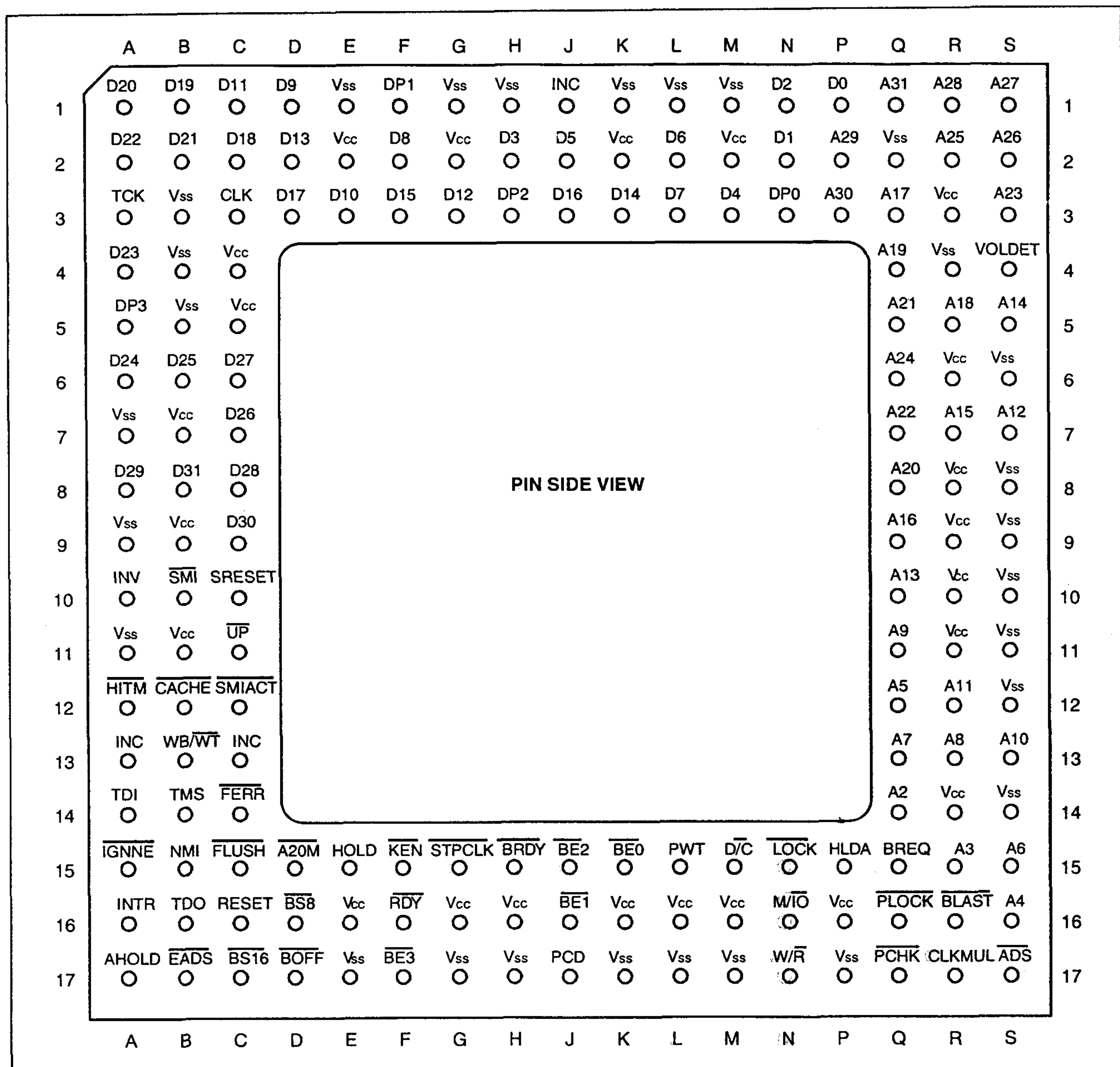
Tabel 7/4.2-1: Overzicht van de leverbare AMD enhanced 486-typen.

Aansluitingen en signalen

Aansluitgegevens

In figuur 7/4.2-2 zijn de aansluitingen van de 168-pens PGA-versie (Pin Grid Array) van de Am486DX te zien. De 486 van Intel (met dezelfde behuizing) heeft geen JTAG-voorzieningen en ook de copy-back en write-back buffers ontbreken. De i486SX is identiek aan de i486DX, maar mist de inwendige FPU. De lijst met op functie gesorteerde pen-nummers (tabel 7/4.2-2) geldt zowel voor de DX als de SX uitvoering van de 486 microprocessor. Bij de Intel-typen zal echter wat meer n.c. worden vermeld.
De 75 MHz en 100 MHz typen van de Am486DX4 zijn bovendien leverbaar in een 208-pens SQFP (Shrink Quad Flat Pack) behuizing voor draagbare toepassingen (figuur 7/4.2-3 en tabel 7/4.2-3).

4.2 80486



Figuur 7/4.2-2: Onderaanzicht (op de aansluitpennen) van de 168-pens PGA-versie van de Am486 microprocessor.

4.2 80486

Address		Data		Control		Test		INC	V_{cc}	V_{ss}
Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin No.	Pin No.	Pin No.
A2	Q-14	D0	P-1	$\overline{A20M}$	D-15	TCK	A-3	A-13	B-7	A-7
A3	R-15	D1	N-2	$\overline{ADS}$	S-17	TDI	A-14	C-13	B-9	A-9
A4	S-16	D2	N-1	AHOLD	A-17	TDO	B-16	J-1	B-11	A-11
A5	Q-12	D3	H-2	$\overline{BE0}$	K-15	TMS	B-14		C-4	B-3
A6	S-15	D4	M-3	$\overline{BE1}$	J-16				C-5	B-4
A7	Q-13	D5	J-2	$\overline{BE2}$	J-15				E-2	B-5
A8	R-13	D6	L-2	$\overline{BE3}$	F-17				E-16	E-1
A9	Q-11	D7	L-3	$\overline{BLAST}$	R-16				G-2	E-17
A10	S-13	D8	F-2	$\overline{BOFF}$	D-17				G-16	G-1
A11	R-12	D9	D-1	$\overline{BRDY}$	H-15				H-16	G-17
A12	S-7	D10	E-3	BREQ	Q-15				K-2	H-1
A13	Q-10	D11	C-1	$\overline{BS8}$	D-16				K-16	H-17
A14	S-5	D12	G-3	$\overline{BS16}$	C-17				L-16	K-1
A15	R-7	D13	D-2	$\overline{CACHE}$	B-12				M-2	K-17
A16	Q-9	D14	K-3	CLK	C-3				M-16	L-1
A17	Q-3	D15	F-3	CLKMUL	R-17				P-16	L-17
A18	R-5	D16	J-3	D/$\overline{C}$	M-15				R-3	M-1
A19	Q-4	D17	D-3	DP0	N-3				R-6	M-17
A20	Q-8	D18	C-2	DP1	F-1				R-8	P-17
A21	Q-5	D19	B-1	DP2	H-3				R-9	Q-2
A22	Q-7	D20	A-1	DP3	A-5				R-10	R-4
A23	S-3	D21	B-2	$\overline{EADS}$	B-17				R-11	S-6
A24	Q-6	D22	A-2	$\overline{FERR}$	C-14				R-14	S-8
A25	R-2	D23	A-4	$\overline{FLUSH}$	C-15					S-9
A26	S-2	D24	A-6	$\overline{HITM}$	A-12					S-10
A27	S-1	D25	B-6	HLDA	P-15					S-11
A28	R-1	D26	C-7	HOLD	E-15					S-12
A29	P-2	D27	C-6	$\overline{IGNNE}$	A-15					S-14
A30	P-3	D28	C-8	INTR	A-16					
A31	Q-1	D29	A-8	INV	A-10					
		D30	C-9	$\overline{KEN}$	F-15					
		D31	B-8	$\overline{LOCK}$	N-15					
				M/$\overline{IO}$	N-16					
				NMI	B-15					
				PCD	J-17					
				$\overline{PCHK}$	Q-17					
				$\overline{PLOCK}$	Q-16					
				PWT	L-15					
				$\overline{RDY}$	F-16					
				RESET	C-16					
				$\overline{SMI}$	B-10					
				$\overline{SMIACT}$	C-12					
				SRESET	C-10					
				$\overline{STPCLK}$	G-15					
				$\overline{UP}$	C-11					
				VOLDET	S-4					
				WB/$\overline{WT}$	B-13					
				W/$\overline{R}$	N-17					

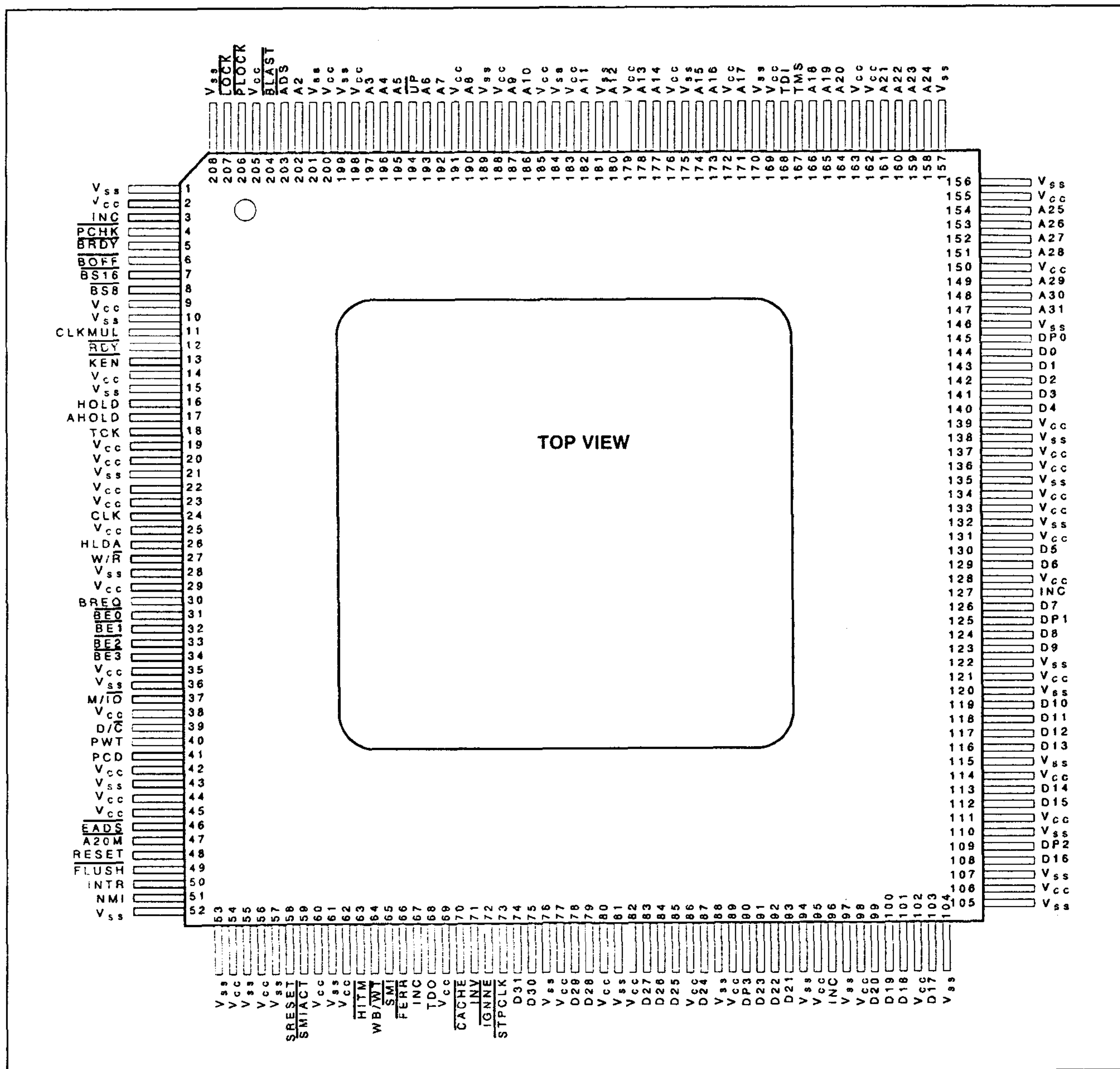
Notes:

VOLDET is connected internally to V_{SS}.

INC = Internal No Connect

Tabel 7/4.2-2: Op functie gesorteerde aansluitingen van de 168-pens PGA-versie van de Am486 processor.

4.2 80486



Figuur 7/4.2-3: Bovenaanzicht van de 208-pens SQFP-versie van de Am486 microprocessor.

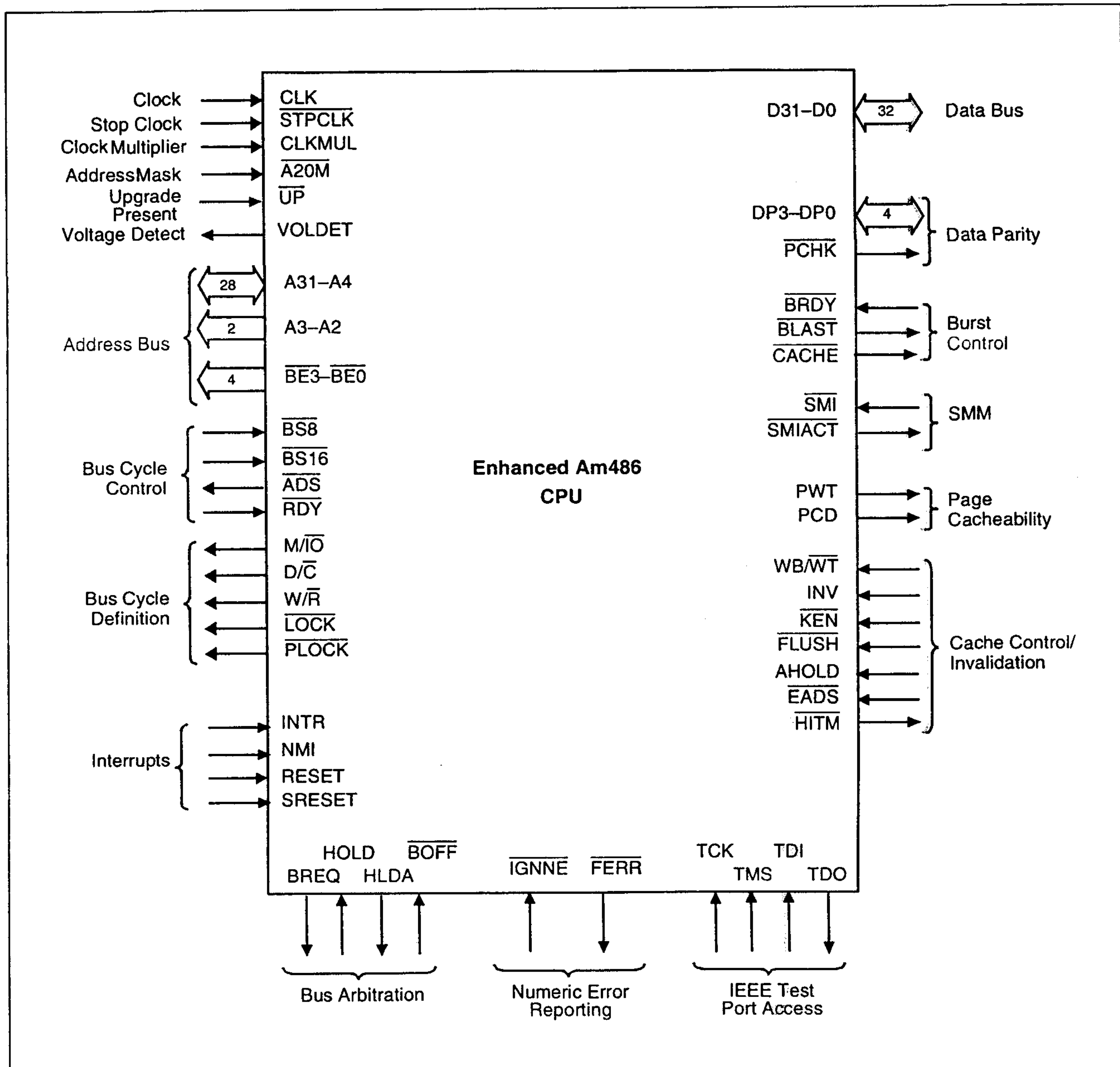
4.2 80486

Address		Data		Control		Test		INC	V_{cc}	V_{ss}
Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin No.	Pin No.	Pin No.
A2	202	D0	144	A20M	47	TCK	18	3	2	1
A3	197	D1	143	ADS	203	TDI	168	67	9	10
A4	196	D2	142	AHOLD	17	TDO	68	96	14	15
A5	195	D3	141	BE0	31	TMS	167	127	19	21
A6	193	D4	140	BE1	32				20	28
A7	192	D5	130	BE2	33				22	36
A8	190	D6	129	BE3	34				23	43
A9	187	D7	126	BLAST	204				25	52
A10	186	D8	124	BOFF	6				29	53
A11	182	D9	123	BRDY	5				35	55
A12	180	D10	119	BREQ	30				38	57
A13	178	D11	118	BS8	8				42	61
A14	177	D12	117	BS16	7				44	76
A15	174	D13	116	CACHE	70				45	81
A16	173	D14	113	CLK	24				54	88
A17	171	D15	112	CLKMUL	11				56	94
A18	166	D16	108	D/C	39				60	97
A19	165	D17	103	DP0	145				62	104
A20	164	D18	101	DP1	125				69	105
A21	161	D19	100	DP2	109				77	107
A22	160	D20	99	DP3	90				80	110
A23	159	D21	93	EADS	46				82	115
A24	158	D22	92	FERR	66				86	120
A25	154	D23	91	FLUSH	49				89	122
A26	153	D24	87	HITM	63				95	132
A27	152	D25	85	HLDA	26				98	135
A28	151	D26	84	HOLD	16				102	138
A29	149	D27	83	IGNNE	72				106	146
A30	148	D28	79	INTR	50				111	156
A31	147	D29	78	INV	71				114	157
		D30	75	KEN	13				121	170
		D31	74	LOCK	207				128	175
				M/IO	37				131	181
				NMI	51				133	184
				PCD	41				134	189
				PCHK	4				136	199
				PLOCK	206				137	201
				PWT	40				139	208
				RDY	12				150	
				RESET	48				155	
				SMI	65				162	
				SRESET	58				163	
				STPCLK	73				169	
				SMIACT	59				172	
				UP	194				176	
				WB/WT	64				179	
				W/R	27				183	
									185	
									188	
									191	
									198	
									200	
									205	

Note:
INC = Internal No Connect

Tabel 7/4.2-3: Op functie gesorteerde aansluitingen van de 208-pens SQFP-versie van de Am486 processor.

4.2 80486



Figuur 7/4.2-4: Logisch symbool van de Am486.

Signalen

Hieronder volgt, in alfabetische volgorde, een korte verklaring van de functies van de gebruikte signalen.

- A20M:
  (input) address bit 20 mask.
- A31-A4/A3-A2:
  (inputs/outputs)/(outputs) de adreslijnen van de microprocessor die samen met de byte-enables BE0-BE3 de fysieke adres- of I/O-ruimte bepalen.
- ADS:
  (output) adres status signaal.
- AHOLD (gemodificeerd):
  (input) address hold request laat een andere busmaster de adresbus gebruiken voor een cache invalidation cyclus.
- BE0-BE3:

4.2 80486

(outputs) byte-enable signalen.
- BLAST (gemodificeerd):
  (output) burst last signaal geeft aan dat met de volgende BRDY de buscyclus klaar is.
- BOFF:
  (input) back off maakt dat de 486 zijn bus bij de volgende clock laat zweven.
- BRDY:
  (input) burst ready geeft aan dat de lopende buscyclus klaar is.
- BREQ:
  (output) internal cycle pending, een door de 486 opgewekte bus-request
- BS8/BS16:
  (inputs) bus size 8 en bus size 16 laten de 486 meerdere buscycli uitvoeren voor schakelingen die geen 32 bit in één cyclus kunnen accepteren.
- CACHE (nieuw):
  (output) internal cacheability geeft aan dat de lopende lees-cyclus gecached kan worden of dat die een burst write-back of copy-back cyclus is.
- CLK (gemodificeerd):
  (input) levert de fundamentele timing.
- CLKMUL (nieuw):
  (input) selecteert de vermenigvuldigingsfactor voor de clock:
  HOOG of open = 3 x CLK, LAAG = 2 x CLK.
- D31-D0:
  (inputs/outputs) datalijnen van de 486.
- D/C:
  (output) data/control lijn.
- DP3-DP0:
  data pariteitspennen.
- EADS (gemodificeerd):
  external address strobe geeft aan dat een geldig extern adres op de adrespennen is gezet (tabel 7/4.2-4).
- FERR:
  (output) floating point error signaal.
- FLUSH:
  (input) cache flush signaal.
- HITM (nieuw):
  (output) hit modified line.
- HLDA:
  (output) hold acknowledge geeft aan dat de besturing van de bus is overgedragen aan een andere lokale busmaster.
- HOLD:
  (input) bus hold request staat toe dat een andere busmaster de microprocessorbus gebruikt.
- IGNNE:
  (input) ignore numeric error signaal.
- INTR:
  (input) maskable interrupt signaal.
- INV (nieuw):
  (input) invalidate, invalideert de toestand van de cache-regel wanneer een externe busmaster wil gaan schrijven.
- KEN:
  (input) cache enable signaal.
- LOCK:
  (output) bus lock signaal.
- M/IO:
  (output) memory/input-output lijn.
- NMI:
  (input) non-maskable interrupt.
- PCD:
  (output) page cache disable signaal.
- PCHK:
  (output) indicatie van de parity status.
- PLOCK (gemodificeerd):
  (output) pseudo lock geeft aan dat voor de lopende operatie meer dan één buscyclus nodig is.
- PWT:
  (output) page write-through signaal.
- RESET:
  (input) zet de 486 in een bekende begintoestand.
- RDY:
  (input) non-burst ready ingang geeft aan dat de lopende buscyclus klaar is.
- SMI (nieuw):
  (input) SMM-interrupt maakt dat de CPU in de System Management Mode gaat (= hoogste interrupt niveau).
- SMIACT (nieuw):
  (output) SMM-interrupt active geeft aan dat de CPU onder SMM control werkt.
- SRESET (nieuw):
  (input) soft reset.

- $\overline{\text{STPCLK}}$ (nieuw):
  (input) stop clock geeft aan dat gevraagd is de CLK-ingang af te schakelen.
- TCK:
  (input) test clock klokt toestand-informatie en data voor de JTAG boundary scan.
- TDI:
  (input) test data seriële ingang voor JTAG instructies en data.
- TDO:
  (output) test data seriële uitgang voor JTAG instructies en data.
- TMS:
  (input) test mode select wordt door de JTAG TAP gedecodeerd om de werking van de testlogika te selecteren.
- $\overline{\text{UP}}$:
  (input) upgrade present signaal.
- VOLDET (nieuw op 168-pen PGA):
  (output) voltage detect voor het bepalen van de voedingsspanning (3 V of 5 V).
- WB/$\overline{\text{WT}}$ (nieuw):
  (input) write-back/write-through.
- W/$\overline{\text{R}}$:
  (output) write/read lijn.

Trigger	$\overline{\text{EADS}}$ First Sampled
AHOLD	Second clock after AHOLD asserted
HOLD	First clock after HLDA asserted
$\overline{\text{BOFF}}$	Second clock after $\overline{\text{BOFF}}$ asserted

Tabel 7/4.2-4: $\overline{\text{EADS}}$ sample-tijd. Het triggersignaal (AHOLD, HOLD of $\overline{\text{BOFF}}$) moet tenminste 1 clock na $\overline{\text{EADS}}$ actief blijven om een goede werking te garanderen.

Functionele beschrijving

Overzicht

De Enhanced Am486 heeft een 32 bit architectuur met een interne memory management unit (MMU), een floating point unit (FPU) en cache geheugen-eenheden. Met de 486 kan alles worden gedaan wat met de 386 ook kan, maar dan beter en sneller. De instructieset van de Enhanced Am486 bevat naast de complete 386 instructieset een aantal nieuwe instructies voor nieuwe, uitgebreide toepassingen. De interne MMU komt volledig overeen met die van de 386, terwijl de 486DX de 387 mathematische coprocessor aan boord heeft.
De Enhanced Am486 heeft een 8 kB cachegeheugen, waardoor veel gebruikte data en code op de chip kan blijven en de bus minder vaak bezet is. Bovendien wordt de cache door middel van de "burst bus" snel gevuld en wordt het aantal benodigde cycli per instructie verminderd door RISC-technieken te gebruiken.

Geheugenbeheer

De geheugenbeheerder (MMU) bestaat uit een segmentatie-eenheid en een paginering-eenheid. Door segmentatie kan de logische adresruimte worden beheerd met behulp van gemakkelijk lokaliseerbare data en code, terwijl efficiënt gezamenlijk gebruik van globale hulpbronnen mogelijk is. De paginering werkt een niveau lager dan de segmentatie en is transparant voor het segmentatieproces. Paginering is niet verplicht en kan door de systeem-software worden uitgeschakeld. Elk segment kan in één of meer 4 kB segmenten worden onderverdeeld. Om een virtueel geheugensysteem te implementeren kan de Enhanced Am486 altijd opnieuw starten na alle segment- en paginafouten.
Het geheugen is georganiseerd in één of meer segmenten met variabele lengte (maximaal 4 GB per segment). Een segment kan vergezeld gaan van attributen die de lokatie, afmetingen, type (bijvoorbeeld stack, code of data) en beveiligingskarakteristieken ervan bevatten. Iedere taak van een 486 processor kan maximaal 16.381 segmenten van elk maximaal 4 GB bevatten. Voor elke taak is dus een maximum van 64 TB virtueel geheugen beschikbaar.

4.2 80486

De segmentatie-eenheid levert beveiliging op vier niveaus voor de isolatie en beveiliging tussen toepassingen en bedrijfssysteem. De door middel van hardware opgelegde beveiliging maakt het ontwerpen van zeer veilige systemen mogelijk.

Bedrijfsmodes

De Enhanced Am486 heeft vier bedrijfsmodes:
- Real Address Mode (Real Mode);
- Virtual 8086 Address Mode (Virtual Mode);
- Protected Address Mode (Protected Mode);
- System Management Mode (SMM).

In de Real Mode (reële mode) werkt de Enhanced Am486 als een zeer snelle 8086 processor. Deze mode wordt hoofdzakelijk gebruikt om de processor in te stellen voor bedrijf in de Protected Mode.
In de Virtual Mode (virtuele mode) lijkt het of de processor zich in de reële mode bevindt, maar hij kan dan de uitgebreide geheugentoegang van de protected mode gebruiken.
De Protected Mode (beveiligde mode) geeft toegang tot de uitgekiende geheugenpaginering en de privileges van de processor.
De System Management Mode (SMM) stelt de ontwerper in staat om nieuwe softwarebestuurde eigenschappen aan computerprodukten toe te voegen, die altijd transparant voor het bedrijfssysteem (OS) en de software-toepassingen zijn. SMM dient alleen door systeem-firmware gebruikt te worden en niet door toepassings-software of algemene systeem-software.

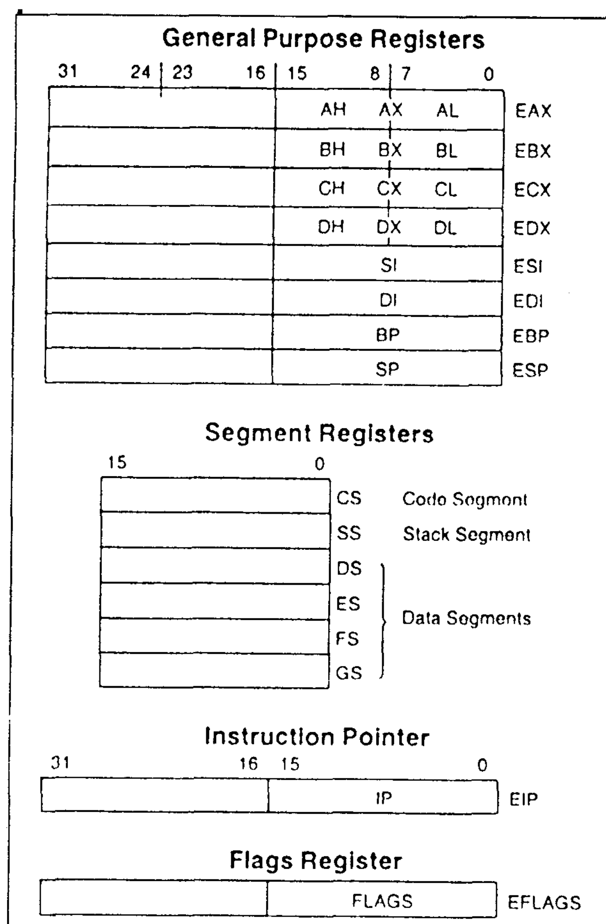
Rekeneenheid

De op de chip aanwezige drijvende komma-eenheid (FPU) werkt parallel met de rekenkundige en logische eenheid en levert rekenkundige instructies voor verschillende numerieke datatypen. Hij voert ingebouwde transcedentale functies, zoals tangens, sinus, cosinus en logaritmische functies uit. De FPU komt volledig overeen met de ANSI/IEEE standaard 754-1985.

De register-set

Inleiding

De Enhanced Am486 microprocessor bevat alle registers die ook in de 386 en 387 zijn opgenomen. De registerset kan worden onderverdeeld in vier groepen:
- Registers voor de basisarchitectuur:
  - algemene registers;
  - instructie-pointer;
  - vlagregister;
  - segmentregisters.



Figuur 7/4.2-5: Registers voor de basisarchitectuur.

- Registers op systeemniveau:
  - besturingsregisters;
  - systeem-adresregisters.
- Floating point-registers:
  - data-registers;

4.2 80486

- tag-woord;
- status-woord;
- instructie- en data-pointers;
- control-woord.

- Debug- en test-registers.

De basis-architectuur- en floating point-registers zijn toegankelijk via toepassingsprogramma's.
De registers op systeemniveau kunnen alleen worden bereikt op privilege-niveau 0 en worden toegepast door het programma op systeem-niveau. De test- en debug-registers zijn ook alleen toegankelijk op privilege-niveau 0.

Registers voor de basisarchitectuur
Figuur 7/4.2-5 toont de registers voor de basisarchitectuur van de 486 processor. Deze registers worden bij het wisselen van een taak automatisch geladen met een nieuwe context. Bij de basisarchitectuur behoren ook zes direct toegankelijke descriptors, die elk een segment van maximaal 4 GB kunnen specificeren. De descriptors worden aangewezen door de selector-waarden die in de segment-registers van de 486 worden geplaatst.
Tijdens de uitvoering van een programma kunnen verschillende selector-waarden worden geladen.

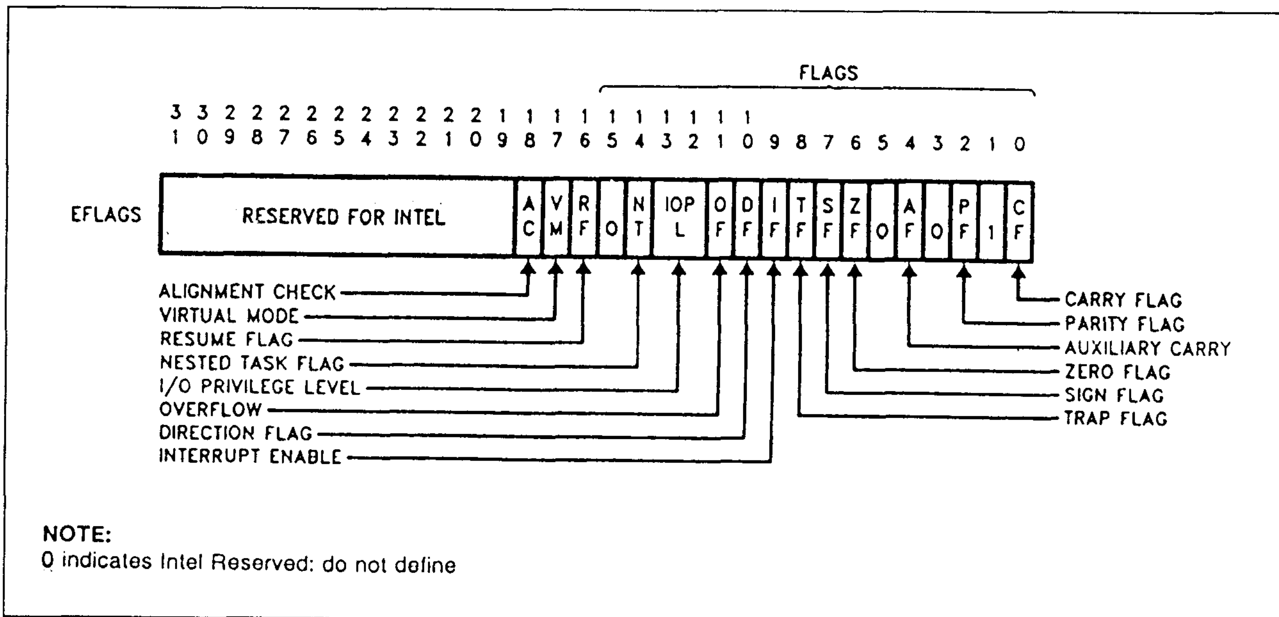
Algemene registers
In de acht 32 bit algemene registers (EAX, EBX, ECX, EDX, ESI, EDI, EBP en ESP) worden data of adres-waarden opgeslagen. Ze zijn geschikt voor data-operands van 1, 8, 16 en 32 bit en bit-velden van 1 tot 32 bit, terwijl adres-operands van 16 en 32 bit worden ondersteund.
Toegang tot de laagste 16 bits van de algemene registers kan apart worden verkregen door gebruik te maken van de 16 bit aanduidingen van de registers (AX, BX, CX, DX, SI, DI, BP en SP). De hoogste 16 bits veranderen niet wanneer apart toegang wordt verkregen tot de laagste 16 bits.
Tenslotte krijgen 8 bit operaties individueel toegang tot het laagste byte (bits 0 tot en met 7) en het hogere byte (bits 8 tot en met 15) van de algemene registers AX, BX, CX en DX. De laagste bytes worden achtereenvolgens AL, BL, CL en DL genoemd en de hogere bytes AH, BH, CH en DH. Door deze individuele toegankelijkheid voor bytes zijn data-operaties flexibeler geworden, maar dit wordt niet gebruikt voor effectieve adresberekeningen

Instructie Pointer
De instructie-pointer (EIP) is een 32 bit register waarin de offset van de volgende uit te voeren instructie wordt opgeslagen. Deze offset is altijd relatief ten opzichte van de basis van het code-segment (CS). De laagste 16 bits (bits 0 tot en met 15) van de EIP bevatten de 16 bit instructie-pointer (IP) die wordt gebruikt voor 16 bit adresseringen.

Vlag Register
Het vlag-register (Flags Register) is een 32 bit register dat EFLAGS wordt genoemd. De gedefinieerde bits en bitvelden in EFLAGS besturen bepaalde operaties en geven de status van de 486 microprocessor aan. De laagste 16 bits (bit 0 tot en met 15) van EFLAGS omvatten het 16 bit register FLAGS dat zeer nuttig is bij de uitvoering van 8086- en 80286-code. In figuur 7/4.2-6 is EFLAGS in zijn geheel te zien.
De EFLAGS bits 1, 3, 5, 15 en 19 tot en met 31 zijn "onbepaald". Wanneer deze bits tijdens de verwerking van een interrupt of bij een PUSHF instructie (push flags onto stack) worden opgeslagen, wordt een logische "1" in bit 1 gezet en nullen in de overige genoemde bits.
Het EFLAGS register in de 486 bevat een nieuw, nog niet eerder benoemd bit: AC. Dit bit bevindt zich tussen de hoogste 16 bits van het register en meldt storingen bij de toegang tot verkeerd uitgerichte data.

AC (Alignment Check, bit 18)
Ter illustratie: goed uitgerichte 4 byte woorden beginnen op adressen die veelvouden van vier zijn.

4.2 80486



Figuur 7/4.2-6: Het 486 vlag-register.

Memory Access	Alignment (Byte Boundary)
Word	2
Dword	4
Single Precision Real	4
Double Precision Real	8
Extended Precision Real	8
Selector	2
48-Bit Segmented Pointer	4
32-Bit Flat Pointer	4
32-Bit Segmented Pointer	2
48-Bit "Pseudo-Descriptor"	4
FSTENV/FLDENV Save Area	4/2 (On Operand Size)
FSAVE/FRSTOR Save Area	4/2 (On Operand Size)
Bit String	4

Tabel 7/4.2-5: Noodzakelijke uitrichtingen voor verschillende data-typen.

Het AC-bit maakt een foutmelding mogelijk als een geheugenverwijzing naar een verkeerd uitgericht adres verwijst. Het melden van uitrichtfouten wordt mogelijk als AC "1" is gemaakt. Een verkeerd uitgericht adres (misaligned address) ontstaat als een woordtoegang plaats vindt op een oneven adres, of als een dubbelwoord wordt gezet op een adres dat niet op een dubbelwoordgrens (dword boundary) staat, of bij een 8 byte verwijzing die niet op een 64 bit woordgrens staat.

Uitricht-foutmeldingen worden alleen gegenereerd door programma's die op privilege-niveau 3 werken. De toestand van het AC-bit wordt genegeerd op de privilege-niveaus 0, 1 en 2. Uitricht-fouten worden gemeld door interrupt 17, met een foutcode 0.
Tabel 7/4.2-5 geeft een overzicht van de vereiste uitrichtingen voor data-typen die in de 486 optreden. Merk op dat verschillende instructies in de 486 microprocessor verkeerd uitgerichte verwijzingen opleveren, ook al zijn de geheugen-adressen zelf goed uitgericht. De SGDT/SIDT (Store Global/Interrupt Descriptor Table) instructie leest/schrijft bijvoorbeeld twee bytes, waarna op de gegeven adressen vier bytes uit een "pseudo-descriptor" worden gelezen/geschreven. De 486 microprocessor zal dan verkeerd uitgerichte adressen genereren, tenzij het adres zich op een 2 mod 4 grens bevindt. De FSAVE en FRSTOR instructies (Floating Point Save en Restore State) wekken verkeerd uitgerichte verwijzingen op voor de helft van de register save/restore cycli. De 486 processor veroorzaakt geen AC-meldingen als het effectieve adres in de instructie de juiste uitrichting heeft.

VM (Virtual 8086 Mode, bit 17)
Het VM-bit zorgt voor de virtuele 8086 mode binnen de beveiligde mode. Als VM "1" wordt gemaakt, terwijl de 486 zich in de beveiligde mode bevindt, schakelt de 486 over naar Virtueel 8086 bedrijf. Hierbij gebeurt het laden van segmenten op dezelfde manier als bij de 8086, maar bij geprivilegeerde opcodes wordt uitzondering 13 gegenereerd. Het VM-bit kan alleen worden gezet in de beveiligde mode door de IRET instructie (als het huidige privilege-niveau 0 is) en door taak-omschakelingen bij elk willekeurig privilege-niveau. Het VM-bit wordt niet beïnvloed door POPF, terwijl PUSHF altijd een "0" in dit bit zet.

RF (Resume Flag, bit 16)
De RF-vlag wordt samen met de debug register-breekpunten gebruikt. Hij wordt aan de randen van de instructies gecheckt voordat de breekpunten worden verwerkt. Wanneer RF "1" is gemaakt, wordt iedere debug foutmelding bij de volgende instructie genegeerd. Na het succesvol beëindigen van een instructie wordt RF dan automatisch gereset, behalve bij de IRET of POPF instructie (en de JMP, CALL en INT instructies die een taakomschakeling veroorzaken). Deze instructies zetten RF op de waarde die door de inhoud van het geheugen wordt gespecificeerd. Aan het einde van de breekpunt service-routine kan de IRET instructie bijvoorbeeld een EFLAG beeld poppen waarbij het RF-bit wordt gezet, terwijl het programma wordt beëindigd op het breekpunt-adres zonder dat een nieuwe breekpunt-foutmelding op dezelfde lokatie wordt gegenereerd.

NT (Nested Task, bit 14)
Deze vlag heeft betrekking op de beveiligde mode. NT wordt op "1" gezet om aan te geven dat de uitvoering van deze taak is genest binnen een andere taak. Hij is gezet als het Task State Segment (TSS) van de actuele taak een geldige terugkoppeling heeft naar de TSS van de vorige taak. Dit bit wordt gezet en gereset door besturingsoverdrachten naar andere taken. De waarde van NT in EFLAGS wordt getest door de IRET instructie om na te gaan of een intertaak return of een intra-taak return moet worden gedaan.

IOPL (Input/Output Privilege Level, bits 12 en 13)
Dit twee bit veld heeft betrekking op de beveiligde mode. IOPL geeft de numeriek maximaal toegestane CPL-waarde (Current Privilege Level) aan om I/O-instructies uit te voeren zonder een uitzondering 13 te veroorzaken of de I/O Permission Bitmap te raadplegen. IOPL geeft ook de maximale CPL-waarde aan waarbij het IF-bit (INTR Enable Flagbit) mag worden veranderd wanneer nieuwe waarden in het EFLAG register worden gepopt. Wanneer de POPF en IRET instructies worden uitgevoerd bij CPL = 0 kan het IOPL-veld worden veranderd. Het IOPL-veld kan altijd worden veranderd door taakomschakelingen wanneer het nieuwe vlagbeeld uit de TSS van de binnenkomende taak wordt geladen.

OF (Overload Flag, bit 11)
OF wordt op "1" gezet als de operatie een overflow met teken veroorzaakt. Dit gebeurt als de operatie een carry/borrow naar het tekenbit (hoogste bit) van het resultaat tot gevolg heeft, maar geen carry/borrow vanuit het hoogste bit of omgekeerd. Voor 8, 16 of 32 bit operaties wordt OF dus gezet naar gelang de overflow op bit 7, 15 of 31.

DF (Direction Flag, bit 10)
DF bepaalt of ESI en/of EDI registers worden verhoogd of verlaagd (post-increment of post-decrement) bij string-operaties. Verhoging treedt op als DF is gereset en verlaging als DF (op 1) is gezet.

IF (INTR Enable Flag, bit 9)
Als de IF-vlag is gezet kunnen externe interrupts worden herkend als ze op de INTR-pen verschijnen. Wanneer IF op "0" is gereset, worden externe interrupts niet gedetecteerd.

4.2 80486

IOPL geeft de maximale CPL-waarde aan waarbij het IF-bit mag worden veranderd wanneer nieuwe waarden in EFLAGS of FLAGS worden gepopt.

TF (Trap Enable Flag, bit 8)
TF regelt de opwekking van de uitzondering 1-sprong bij het stap-voor-stap doorlopen van de code. Wanneer TF op "1" is gezet wekt de 486 microprocessor een uitzondering 1-sprong op na het uitvoeren van de volgende instructie. Bij geresette TF treden uitzondering 1-sprongen alleen op als een functie van de breekpunt-adressen die in de debug-registers DR0 tot en met DR3 zijn geladen.

SF (Sign Flag, bit 7)
SF wordt op "1" gezet als het hoogste bit van het resultaat is gezet (en omgekeerd). Voor 8, 16 en 32 bit operaties komt SF overeen met de toestand van respectievelijk bit 7, 15 of 31.

ZF (Zero Flag, bit 6)
ZF wordt op "1" gezet als alle bits van een resultaat "0" zijn. In andere gevallen is ZF gereset.

AF (Auxiliary Carry Flag, bit 4)
De AF-vlag wordt gebruikt om het optellen en aftrekken van gepakte BCD hoeveelheden gemakkelijker te maken. AF wordt op "1" gezet als de operatie een carry uit bit 3 (optelling) of een borrow in bit 3 (aftrekking) tot gevolg had. AF wordt alleen beïnvloed door carry uit of borrow in bit 3, onafhankelijk van de totale lengte van de operand (8, 16 of 32 bit).

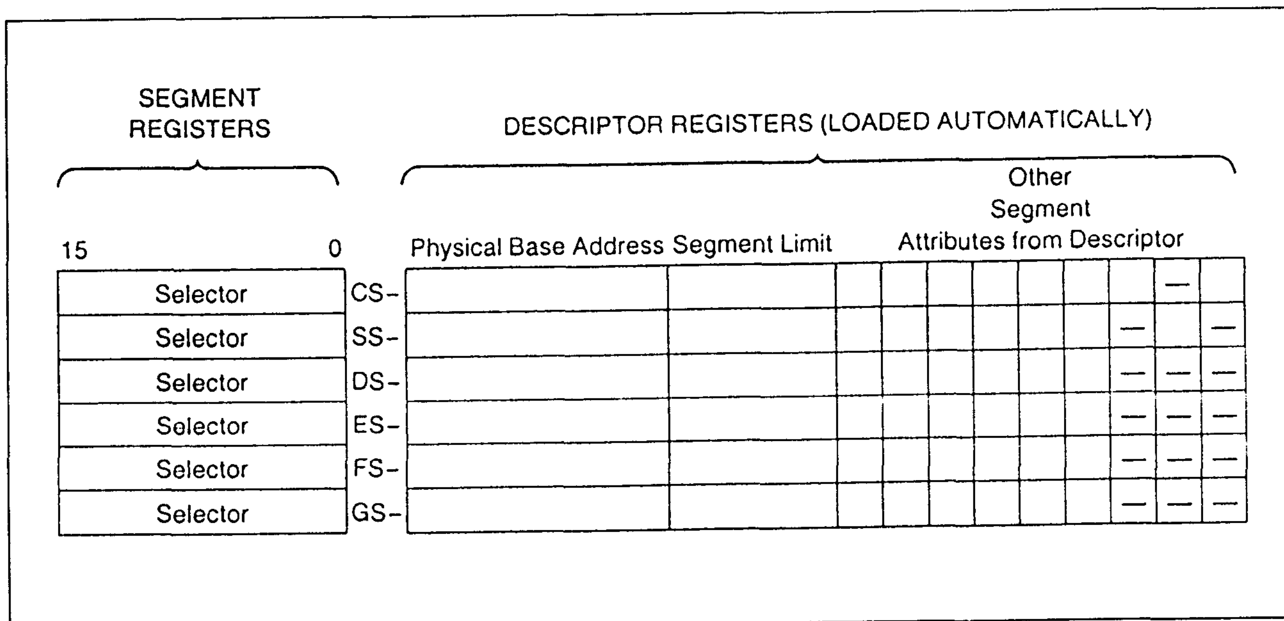
PF (Parity Flag, bit 2)
PF wordt op "1" gezet als de laagste acht bits van de operatie een even aantal "enen" bevat (= even pariteit).
Bij een oneven pariteit wordt PF gereset. PF is een functie van alleen de laagste acht bits, onafhankelijk van de totale lengte van de operand.

CF (Carry Flag, bit 0)
CF wordt "1" als de operatie in een carry uit (optelling) of borrow in (aftrekking) van het hoogste bit resulteert. In andere gevallen is CF gereset. Voor 8, 16 en 32 bit operaties wordt CF dus gezet naar gelang de waarde van bit 7, 15 of 31.

Segment Registers
Voor de opslag van segment-selectorwaarden die de actuele adresseerbare geheugensegmenten identificeren, worden zes 16 bit segmentregisters gebruikt. In de beveiligde mode kan de afmeting van elk segment variëren van één byte tot de gehele lineaire en fysieke adresruimte van de machine: 4 GB (2^{32} byte). In de reële adresseringsmode is de maximale segmentgrootte gefixeerd op 64 kB (2^{16} byte).
De zes adresseerbare segmenten worden gedefinieerd door de segment-registers CS, SS, DS, ES, FS en GS.
De selector in CS geeft het actuele codesegment aan, die in SS het actuele stacksegment en de overige de actuele datasegmenten.

Segment Descriptor Cache Registers
Hoewel de segment descriptor cacheregisters van buitenaf niet zichtbaar zijn, is het zeer nuttig te begrijpen waarvoor ze dienen.
Bij elk segmentregister dat de programmeur kan zien behoort een voor hem onzichtbaar descriptor cache-register (zie figuur 7/4.2-7). Ieder descriptor cache-register bevat een 32 bit basisadres, een 32 bit segmentlimiet en de overige noodzakelijke segmentattributen.
Wanneer een selectorwaarde in een segmentregister wordt geladen, wordt het bijbehorende descriptor cache-register automatisch bijgewerkt met de juiste informatie. In de reële adresseringsmode wordt alleen het basisadres bijgewerkt (door de selectorwaarde vier bit naar links te schuiven), aangezien de segmentlimiet en de attributen voor de reële mode vastliggen.

4.2 80486



Figuur 7/4.2-7: *Segmentregisters en bijbehorende descriptor cache-registers van de 486 microprocessor.*

In de beveiligde mode worden het basisadres, de limiet en de attributen allemaal ververst door de inhoud van de segment descriptor die is geïndexeerd door de selector. Telkens wanneer een geheugenreferentie optreedt, wordt het bij het segment behorende descriptor cache-register automatisch betrokken bij deze geheugenreferentie. Het 32 bit segment-basisadres wordt een component voor de berekening van het lineaire adres, de 32 bit limiet wordt gebruikt door de limiet-check operatie en de attributen worden afgewogen tegen de gevraagde soort geheugenreferentie.

Registers op systeemniveau

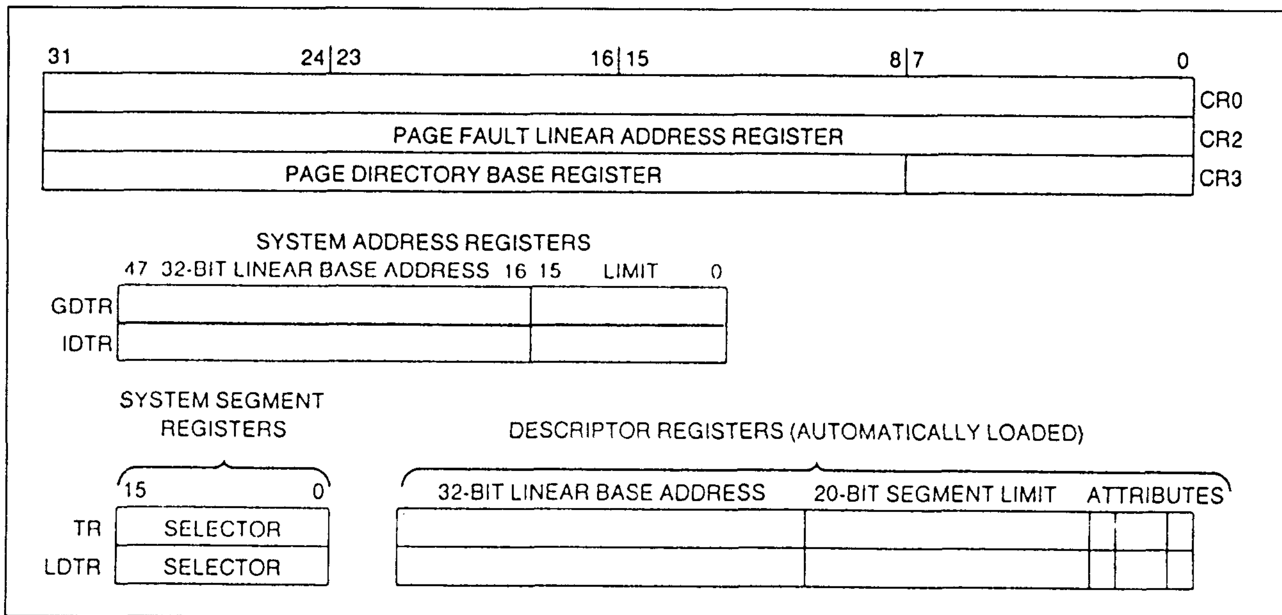
De registers op systeemniveau regelen de werking van de op de chip aanwezige cache, de drijvende komma-eenheid (FPU) en de segmentatie- en paginerings-mechanismen. Deze registers zijn alleen toegankelijk voor programma's die op privilege-niveau 0 draaien (het hoogste privilege-niveau). De registers op systeemniveau omvatten drie besturingsregisters en vier segmentatie-basisregisters (zie figuur 7/4.2-8). De besturingsregisters zijn CR0, CR2 en CR3 (CR1 is gereserveerd voor toekomstige processoren). De vier segmentatie-basisregisters zijn het globale descriptor tabel-register (GDTR), het interrupt descriptor tabel-register (IDTR), het lokale descriptor tabel-register (LDTR) en het taak status segment-register (TR).

Control Register 0 (CR0)

CR0 heeft 10 bits voor besturing en status (figuur 7/4.2-9). De 486 processor heeft in CR0 vijf nieuw benoemde bits: CD, NW, AM, WP en NE. De functie van de CR0-bits kunnen in de volgende categorieën worden ingedeeld:

- 486 bedrijfsmodes:
  PG (Paging Enable), PE (Protection Enable) (tabel 7/4.2-6).
- on-chip Cache-besturingsmodes:
  CD (Cache Disable), NW (Not Write-Through) (tabel 7/4.2-7).
- on-chip FPU-besturing:
  TS (Task Switched), EM (Emulate Coprocessor), MP (Monitor Coprocessor), NE (Numerics Exception) (tabel 7/4.2-8).
- besturing van de uitricht-check:
  AM (Alignment Mask)
- supervisor Write Protect:
  WP (Write Protect).

4.2 80486



Figuur 7/4.2-8: Registers op systeemniveau.

PG	PE	Mode
0	0	REAL Mode. Exact 8086 semantics, with 32-bit extensions available with prefixes.
0	1	Protected Mode. Exact 80286 semantics, plus 32-bit extensions through both prefixes and "default" prefix setting associated with code segment descriptors. Also, a sub-mode is defined to support a virtual 8086 within the context of the extended 80286 protection model.
1	0	UNDEFINED. Loading CR0 with this combination of PG and PE bits will raise a GP fault with error code 0.
1	1	Paged Protected Mode. All the facilities of Protected mode, with paging enabled underneath segmentation.

Tabel 7/4.2-6: 486 bedrijfsmodes.

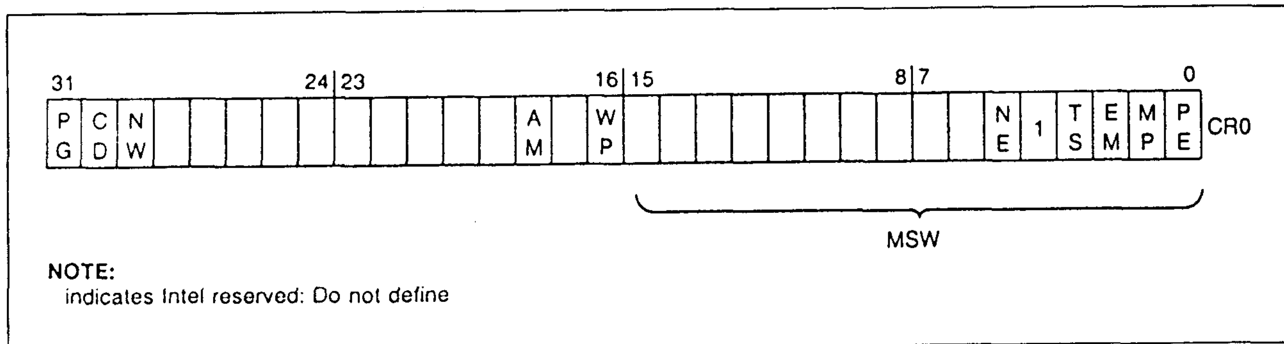
CD	NW	Operating Mode
1	1	Cache fills disabled, write-through and invalidates disabled.
1	0	Cache fills disabled, write-through and invalidates enabled.
0	1	INVALID. If CR0 is loaded with this configuration of bits, a GP fault with error code is raised.
0	0	Cache fills enabled, write-through and invalidates enabled.

Tabel 7/4.2-7: On-chip Cache-besturingsmodes.

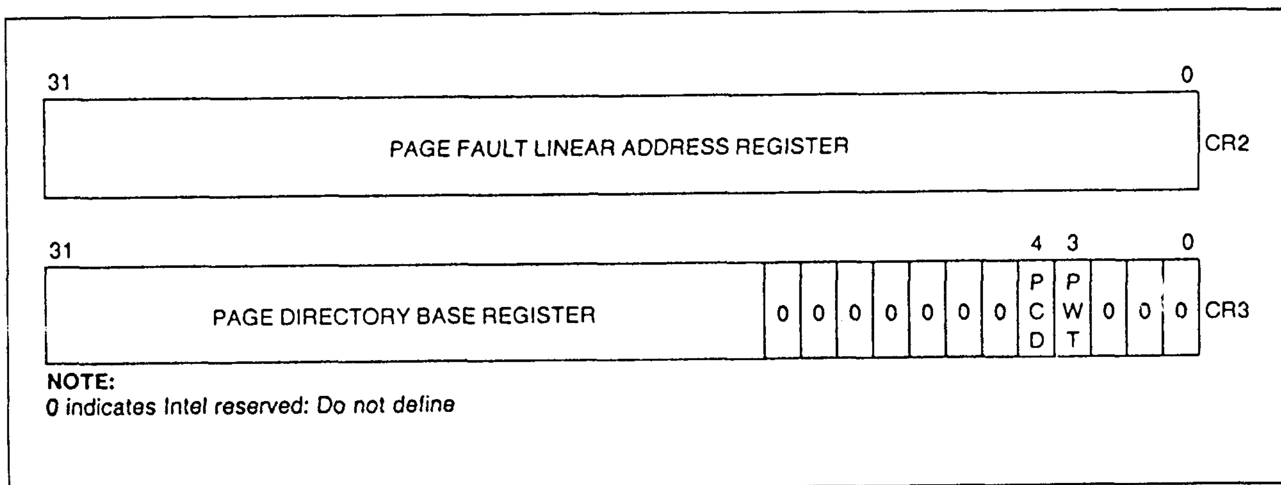
CR0 BIT			Instruction Type	
EM	TS	MP	Floating-Point	Wait
0	0	0	Execute	Execute
0	0	1	Execute	Execute
0	1	0	Trap 7	Execute
0	1	1	Trap 7	Trap 7
1	0	0	Trap 7	Execute
1	0	1	Trap 7	Execute
1	1	0	Trap 7	Execute
1	1	1	Trap 7	Trap 7

Tabel 7/4.2-8: On-chip besturing van de drijvende-komma eenheid (FPU).

4.2 80486



Figuur 7/4.2-9: De indeling van Control Register 0.



Figuur 7/4.2-10: De Control-registers 2 en 3.

Control Register 1 (CR1)

CR1 is gereserveerd voor toekomstige microprocessoren.

Control Register 2 (CR2)

CR2 bevat het 32 bit lineaire adres dat de laatste pagina-foutmelding tot gevolg had. De foutcode die op de stack van de pagina-fout-routine wordt gezet (wanneer die is aangeroepen) levert aanvullende statusinformatie.

Control Register 3 (CR3)

CR3 (zie ook figuur 7/4.2-10) bevat het fysieke basisadres van de Page Directory Tabel. De page directory van de 486 is altijd pagina-uitgericht (4 kB uitrichting). Deze uitrichting komt tot stand door alleen de bits 20 tot en met 31 op te slaan in CR3.

In de 486 bevat CR3 twee nieuwe bits: page write-through (PWT, bit 3) en page cache disable (PCD, bit 4). De page table entry (PTE) en page directory entry (PDE) bevatten ook PWT en PCD bits. Met PWT en PCD wordt de eventuele opslag van een pagina in cache geregeld. Wanneer toegang tot een pagina in extern geheugen nodig is, wordt de toestand van PWT en PCD op de betreffende pennen gezet. PWT en PCD kunnen afkomstig zijn van CR3, de PTE of de PDE. Er zijn twee omstandigheden waaronder PWT en PCD uit CR3 komen: als de paginering wordt gesperd (PG=0 in CR0) of wanneer de PDE wordt herschreven.
Wanneer de waarden in CR3 veranderen als gevolg van een taakomschakeling door een Taak-Toestand Segment (TSS) heen, of wanneer CR3 expliciet met een bepaalde

waarde wordt geladen, worden alle gecachte page table entries in de Translation Lookaside Buffer (TLB) ongeldig.
Het basisadres van de page directory in CR3 is een fysiek adres. De page directory kan worden uitgepagineerd bij uitstel van zijn bijbehorende taak, maar het bedrijfssysteem moet ervoor zorgen dat de page directory resident is in fysiek geheugen voordat de taak wordt verzonden.

Systeem-adres registers

Er zijn vier registers gedefinieerd als referentie voor de tabellen of segmenten die worden ondersteund door het beveiligingsmodel van de 80286, 80386 en 80486. Deze tabellen of segmenten zijn:

- GDT (Global Descriptor Table);
- IDT (Interrupt Descriptor Table);
- LDT (Local Descriptor Table);
- TSS (Task State Segment).

De adressen van deze tabellen en segmenten zijn opgeslagen in speciale registers: de Systeem Adres Registers en de Systeem Segment Registers (zie ook figuur 7/4.2-8). Deze registers worden GDTR, IDTR, LDTR en TR genoemd.

Systeem Adres Registers GDTR en IDTR

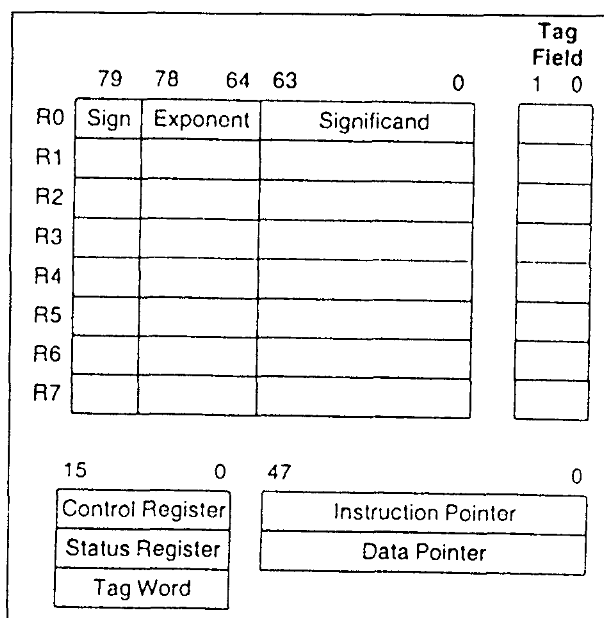
Het Global Descriptor Table Register (GDTR) bevat het 32 bit lineaire basisadres en het Interrupt Descriptor Table Register (IDTR) de 16 bit limiet van de GDT en de IDT. Aangezien de GDT- en IDT-segmenten globaal zijn voor alle taken in het systeem, worden GDT en IDT gedefinieerd door 32 bit adressen (onderworpen aan pagina-omzetting als pagineren is enabled) en 16 bit limietwaarden.

Systeem
Segment Registers LDTR en TR

In het Local Descriptor Table Register (LDTR) wordt de 16 bit selector voor de LDT opgeslagen en in het TR de TSS descriptor. Aangezien de LDT- en TSS-segmenten taak-specifieke segmenten zijn, worden LDT en TSS gedefinieerd door selectorwaarden die in de systeem-segmentregisters zijn opgeslagen. Bij elk systeem-segmentregister behoort een voor de programmeur onzichtbaar segment-descriptorregister.

Floating Point registers

In figuur 7/4.2-11 is de registerset voor de drijvende komma te zien. De FPU bevat acht data-registers, een tag-woord, een besturingsregister, een status-register, een instructie-pointer en een data-pointer. De werking van de drijvende komma-eenheid in de 486 is precies hetzelfde als de 387 mathematische coprocessor. Software die voor de 387 coprocessor is geschreven zal zonder aanpassingen met de FPU in de 486 kunnen samenwerken.

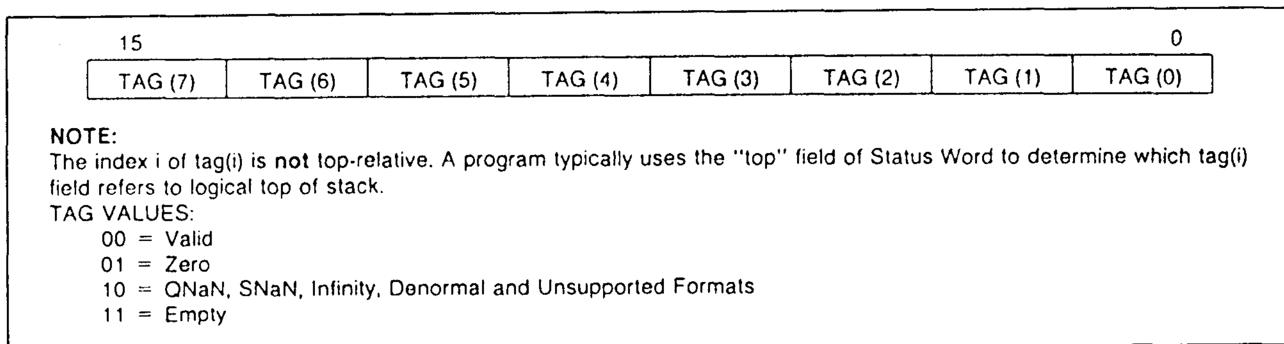


Figuur 7/4.2-11: Floating Point Registers.

Data Registers

Drijvende komma-berekeningen maken in de 486 gebruik van de FPU data-registers. Deze acht 80 bit registers hebben evenveel capaciteit als twintig 32 bit registers. Elk dataregister is in velden onderverdeeld die overeenkomen met het extended-precision data-type van de FPU.

4.2 80486



Figuur 7/4.2-12: Het FPU Tag woord.

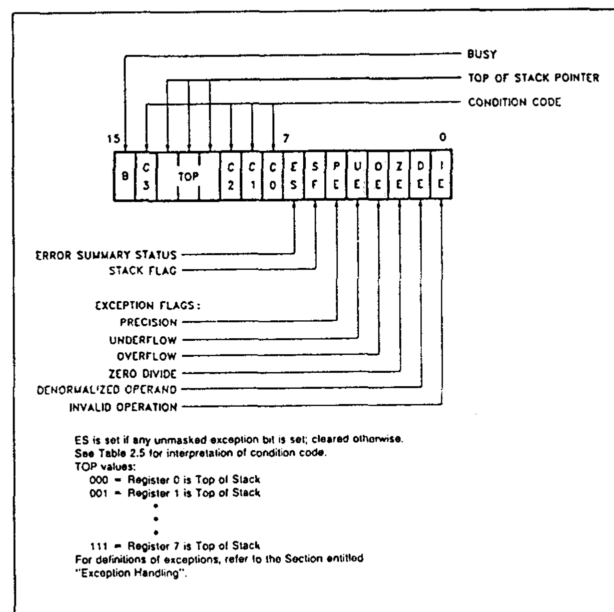
De registerset van de FPU kan worden bereikt als een stack (met instructies die bovenop één of twee stack-elementen werken) of als een gefixeerde registerset (met instructies die op expliciet aangewezen registers werken). Het TOP-veld in het status-woord identificeert het lopende top-of-stack register. Een PUSH operatie verlaagt TOP met één en laadt een waarde in het nieuwe top-register. Een POP operatie slaat de waarde uit het huidige top-register op en verhoogt TOP met één. Net als andere 486 stacks in het geheugen groeit de FPU-stack neer naar lager geadresseerde registers toe.
Instructies kunnen de data-registers impliciet of expliciet adresseren. Veel instructies werken op de TOP van de stack in de registers. Deze instructies adresseren het register waarnaar TOP wijst impliciet. Andere instructies maken expliciete aanwijzing (gerelateerd aan TOP) van het te gebruiken register mogelijk.

Tag Woord

Het tag-woord markeert de inhoud van elk numeriek data-register (zie figuur 7/4.2-12). Elke twee bit tag vertegenwoordigt één van de acht data-registers. Het tag-woord dient om de prestaties van de FPU en het gebruik van de stack te optimaliseren door lege en niet-lege register-lokaties van elkaar te onderscheiden. Het stelt ook uitzonderingshandlers in staat om de inhoud van een stack-lokatie te checken zonder dat het nodig is de actuele data te decoderen.

Status Woord

Het 16 bit status-woord geeft de toestand van de FPU als geheel weer. Het statuswoord (figuur 7/4.2-13) bevindt zich in het statusregister.



Figuur 7/4.2-13: FPU status-woord.

Het B bit (Busy, bit 15) is hierin opgenomen om compatibel te zijn met de 8087 en komt overeen met het ES bit (bit 7) van het status-woord. Bits 11, 12 en 13 wijzen het FPU-register aan dat zich op dat ogenblik bovenaan de stack bevindt. De vier numerieke conditiecode-bits (C0 tot en met C3) komen overeen met de vlaggen in EFLAGS.

4.2 80486

Condition Code				Interpretation after FPREM and FPREM1	
C2	C3	C1	C0		
1	X	X	X	Incomplete Reduction: further interaction required for complete reduction	
0	Q1	Q0	Q2	Q MOD8	Complete Reduction: C0, C3, C1 contain three least significant bits of quotient
	0	0	0	0	
	0	1	0	1	
	1	0	0	2	
	1	1	0	3	
	0	0	1	4	
	0	1	1	5	
	1	0	1	6	
	1	1	1	7	

Tabel 7/4.2-9: Interpretatie van de conditie-code na FPREM en FPREM1 instructies.

Order	C3	C2	C0
TOP > Operand	0	0	0
TOP < Operand	0	0	1
TOP = Operand	1	0	0
Unordered	1	1	1

Tabel 7/4.2-10: Conditie-code als gevolg van vergelijking.

Deze bits worden op orde gebracht door instructies die rekenkundige operaties uitvoeren om de uitkomst weer te geven. De invloed van deze instructies op de conditiecodes zijn te zien in de tabellen 7/4.2-9, -10, -11 en -12.

Bit 7 is het Error Summary (ES) statusbit dat wordt gezet als een niet-gemaskeerd uitzonderingsbit (bits 0 tot en met 5 in het statuswoord) is gezet; ES is in alle andere gevallen gecleared. Wanneer ES is gezet, wordt het $\overline{FERR}$ (Floating point ERRor) signaal afgegeven. Bit 6 is de Stack Flag (SF), waarmee foute operaties als gevolg van stack overflow of underflow kunnen worden gesignaleerd. Als SF is gezet, geeft bit 9 (C1) aan of het een overflow (C1=1) of een underflow (C1=0) betreft.

4.2 80486

Instruction	C0 (S)	C3 (Z)	C1 (A)	C2 (C)
FPREM, FPREM1 (see Table 2.3)	Three least significant bits of quotient Q2	Q0	Q1 or O/U #	Reduction 0 = complete 1 = incomplete
FCOM, FCOMP, FCOMPP, FTST, FUCOM, FUCOMP, FUCOMPP, FICOM, FICOMP	Result of comparison (see Table 2.7)		Zero or O/U #	Operand is not comparable (Table 2.7)
FXAM	Operand class (see Table 2.8)		Sign or O/U #	Operand class (Table 2.8)
FCHS, FABS, FXCH, FINCTOP, FDECTOP, Constant loads, FXTRACT, FLD, FILD, FBLD, FSTP (ext real)	UNDEFINED		Zero or O/U #	UNDEFINED
FIST, FBSTP, FRNDINT, FST, FSTP, FADD, FMUL, FDIV, FDIVR, FSUB, FSUBR, FSCALE, FSQRT, FPATAN, F2XM1, FYL2X, FYL2XP1	UNDEFINED		Roundup or O/U #	UNDEFINED
FPTAN, FSIN FCOS, FSINCOS	UNDEFINED		Roundup or O/U #, undefined if C2 = 1	Reduction 0 = complete 1 = incomplete
FLDENV, FRSTOR	Each bit loaded from memory			
FINIT	Clears these bits			
FLDCW, FSTENV, FSTCW, FSTSW, FCLEX, FSAVE	UNDEFINED			

O/U # — When both IE and SF bits of status word are set, indicating a stack exception, this bit distinguishes between stack overflow (C1 = 1) and underflow (C1 = 0).

Reduction — If FPREM or FPREM1 produces a remainder that is less than the modulus, reduction is complete. When reduction is incomplete the value at the top of the stack is a partial remainder, which can be used as input to further reduction. For FPTAN, FSIN, FCOS, and FSINCOS, the reduction bit is set if the operand at the top of the stack is too large. In this case the original operand remains at the top of the stack.

Roundup — When the PE bit of the status word is set, this bit indicates whether the last rounding in the instruction was upward.

UNDEFINED — Do not rely on finding any specific value in these bits.

Tabel 7/4.2-11: Interpretatie van de FPU conditie-code.

(wordt vervolgd)

4.2 80486

C3	C2	C1	C0	Value at TOP
0	0	0	0	+ Unsupported
0	0	0	1	+ NaN
0	0	1	0	− Unsupported
0	0	1	1	− NaN
0	1	0	0	+ Normal
0	1	0	1	+ Infinity
0	1	1	0	− Normal
0	1	1	1	− Infinity
1	0	0	0	+ 0
1	0	0	1	+ Empty
1	0	1	0	− 0
1	0	1	1	− Empty
1	1	0	0	+ Denormal
1	1	1	0	− Denormal

Tabel 7/4.2-12: Bepaling van de conditie-code door de operand klasse.

Uitzonderingsvlaggen

Tabel 7/4.2-13 laat de zes uitzonderingsvlaggen in bits 0 tot en met 5 van het status-woord zien. Deze bits worden gezet om aan te geven dat de FPU een uitzondering heeft gedetecteerd bij het uitvoeren van een instructie. De zes uitzonderingsvlaggen in het status-woord kunnen individueel worden gemaskeerd door de maskeerbits in het FPU besturings-woord. In tabel 7/4.2-13 zijn niet alleen de uitzonderingscondities en de oorzaken ervan in volgorde van prioriteit te zien, maar ook de actie die door de FPU wordt ondernomen als de betreffende uitzonderingsvlag is gemaskeerd.

Een uitzondering die niet door het besturings-woord wordt gemaskeerd zal drie dingen tot gevolg hebben: de overeenkomstige uitzonderingsvlag in het status-woord wordt gezet, het ES bit in het status-woord wordt gezet en het $\overline{FERR}$ uitgangssignaal verschijnt. Wanneer de 486 processor probeert nog een floating point of een WAIT instructie uit te voeren, treedt uitzondering 16 op of een externe interrupt (als NE=1 in besturingsregister 0). De uitzonderingsconditie moet worden opgelost via een interrupt service-routine. De FPU bewaart het adres van de floating point instructie die de uitzondering veroorzaakte en het adres van elke willekeurige geheugen-operand die deze instructie nodig had in de instructie- en data pointers.

Exception	Cause	Default Action (if exception is masked)
Invalid Operation	Operation on a signaling NaN, unsupported format, indeterminate form (0 * ∞, 0/0, (+∞) + (−∞), etc.), or stack overflow/underflow (SF is also set).	Result is a quiet NaN, integer indefinite, or BCD indefinite
Denormalized Operand	At least one of the operands is denormalized, i.e., it has the smallest exponent but a nonzero significand.	Normal processing continues
Zero Divisor	The divisor is zero while the dividend is a noninfinite, nonzero number.	Result is ∞
Overflow	The result is too large in magnitude to fit in the specified format.	Result is largest finite value or ∞
Underflow	The true result is nonzero but too small to be represented in the specified format, and, if underflow exception is masked, denormalization causes loss of accuracy.	Result is denormalized or zero
Inexact Result (Precision)	The true result is not exactly representable in the specified format (e.g., 1/3); the result is rounded according to the rounding mode.	Normal processing continues

Tabel 7/4.2-13: FPU uitzonderingen.

4.2 80486

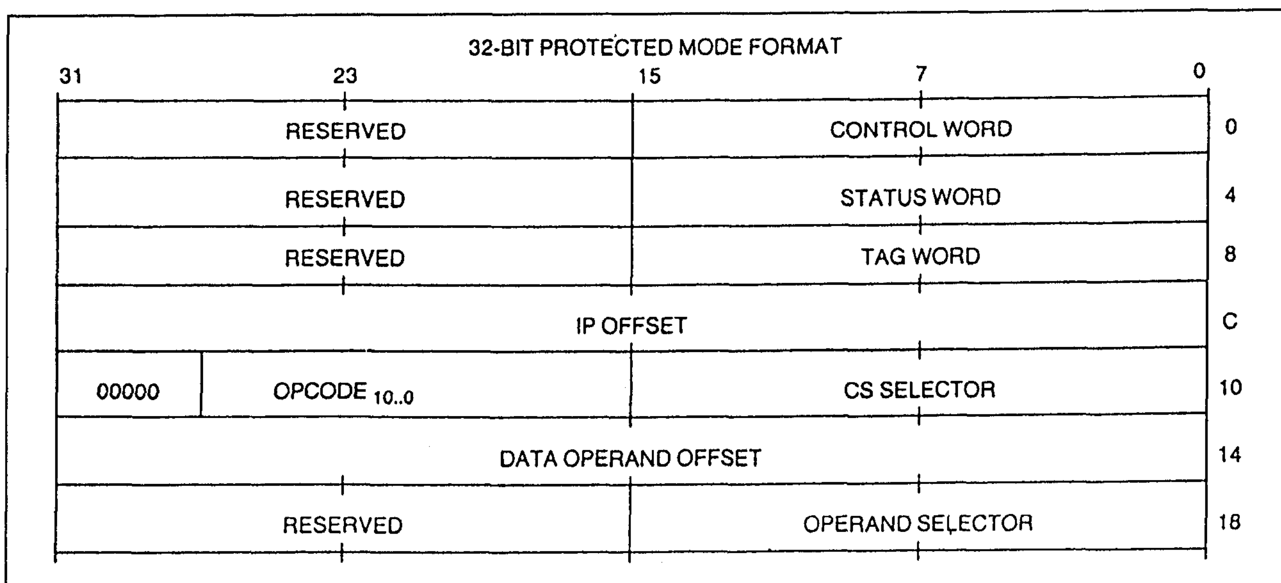
Instructie- en Data-Pointers

Omdat de FPU parallel met de ALU werkt kan het voorkomen dat door de FPU gedetecteerde fouten worden gerapporteerd nadat de ALU de floating point instructie heeft uitgevoerd die de fout veroorzaakte. Om identificatie van de falende numerieke instructie mogelijk te maken bevat de 486 processor twee pointer registers die de adressen leveren van de falende numerieke instructie en eventueel de bijbehorende numerieke geheugen-operand. De instructie- en data pointers kunnen voor zelf geschreven fout-routines worden gebruikt. Deze registers worden bereikt door de FLDENV (load environment), FSTENV (store environment), FSAVE (save state) en FRSTOR (restore state) instructies. Telkens wanneer de 486 een nieuwe floating point instructie decodeert, worden de instructie (inclusief eventuele prefixes), het (eventuele) adres van de operand en de opcode bewaard. De instructie- en data-pointers verschijnen in een formaat (vier mogelijkheden) dat afhankelijk is van de bedrijfsmode van de 486 microprocessor (beveiligde mode of real-address mode) en van de grootte van de operand (32 bit of 16 bit operand).

Wanneer de 486 in de virtuele 86-mode werkt, worden de real address mode formaten gebruikt. De vier formaten worden getoond in de figuren 7/4.2-14 tot en met -17. De floating point instructies FLDENV, FSTENV, FSAVE en FRSTOR worden gebruikt om deze waarden van en naar geheugen over te brengen. Merk op dat de waarde van de data pointer onbepaald is als de voorafgaande floating point instructie geen geheugen-operand heeft.

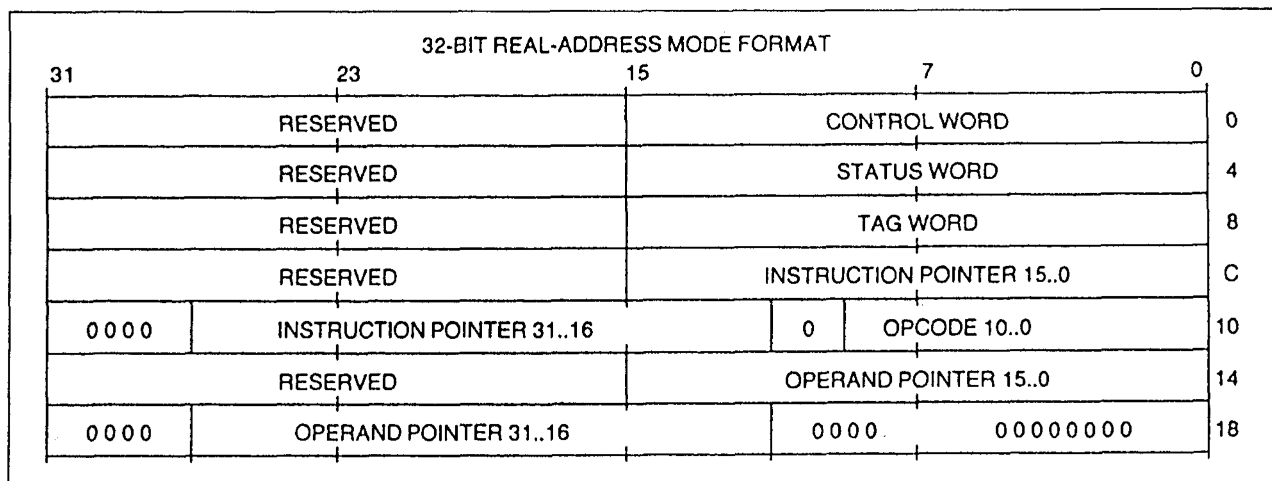
FPU besturings-woord

De FPU kan op verschillende manieren werken, afhankelijk van het besturings-woord dat uit het geheugen in het besturingsregister wordt geladen. Het formaat en de codering van de velden in het besturingswoord zijn te zien in figuur 7/4.2-18. Het laagste byte van het FPU besturings-woord configureert de maskering van de FPU fouten en uitzonderingen. De bits 0 tot en met 5 van het besturings-woord bevatten individuele maskeringen voor elk van de zes uitzonderingen die de FPU herkent. Het hoogste byte van het besturings-woord bepaalt de bedrijfsmode van de FPU, inclusief precisie en afronding.

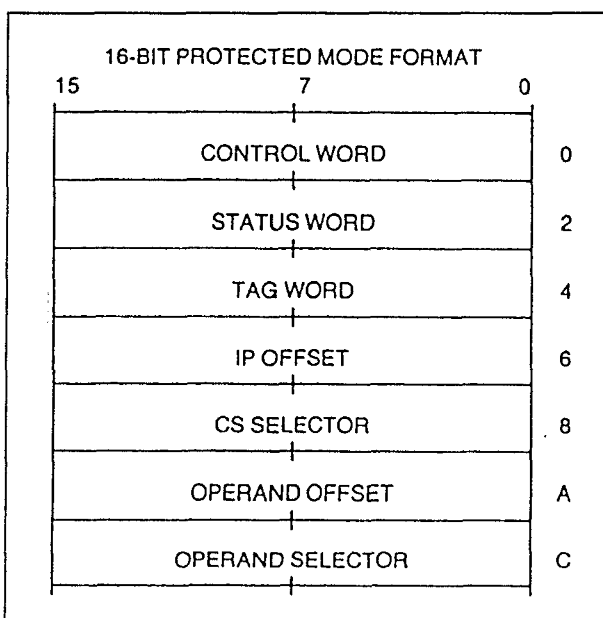


Figuur 7/4.2-14: Vorm van de Protected Mode FPU Instructie- en Data Pointer in het geheugen (32 bit formaat).

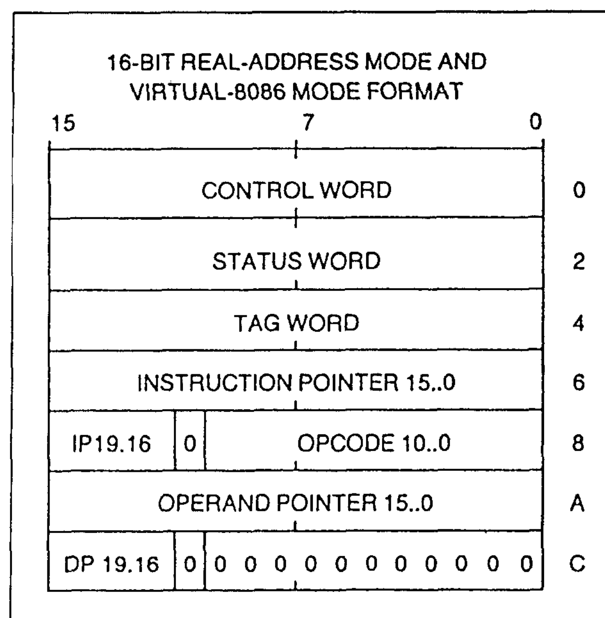
4.2 80486



Figuur 7/4.2-15: Vorm van de Real Mode FPU Instructie- en Data-Pointer in het geheugen (32 bit formaat).

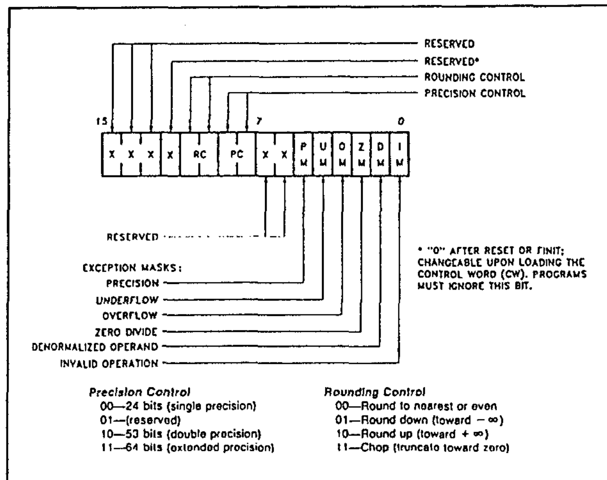


Figuur 7/4.2-16: Vorm van de Protected Mode FPU Instructie- en Data-Pointer in het geheugen (16 bit formaat).



Figuur 7/4.2-17: Vorm van de Real Mode FPU Instructie- en Data-Pointer in het geheugen (16 bit formaat).

4.2 80486



Figuur 7/4.2-18: Het FPU besturings-woord (FPU control word).

Debug- en Test-registers

In figuur 7/4.2-19 wordt een overzicht gegeven van de debug- en test-registers van de 486 microprocessor.

Debug Registers

LINEAR BREAKPOINT ADDRESS 0	DR0
LINEAR BREAKPOINT ADDRESS 1	DR1
LINEAR BREAKPOINT ADDRESS 2	DR2
LINEAR BREAKPOINT ADDRESS 3	DR3
Intel Reserved Do Not Define	DR4
Intel Reserved Do Not Define	DR5
BREAKPOINT STATUS	DR6
BREAKPOINT CONTROL	DR7

Test Registers

CACHE TEST DATA	TR3
CACHE TEST STATUS	TR4
CACHE TEST CONTROL	TR5
TLB TEST CONTROL	TR6
TLB TEST STATUS	TR7

TLB = Translation Lookaside Buffer

Figuur 7/4.2-19: Debug- en test-registers.

Debug-registers

De programmeur kan over zes debug-registers beschikken die on-chip ondersteuning voor het foutzoeken leveren. De debug-registers DR0 tot en met DR3 (zie figuur 7/4.2-19) specificeren de vier lineaire breekpunten.
Het debug-besturings-register DR7 wordt gebruikt om de breekpunten te zetten en het debug-status-register DR6 laat de huidige toestand van de breekpunten zien.

Test-registers

Er zijn vijf test-registers, waarvan TR6 en TR7 dienen om het testen van de Translation Lookaside Buffer te regelen.
Met TR3, TR4 en TR5 kan de on-chip cache worden getest.

Opmerking

De toegang tot de registers verschilt in de Real en de beveiligde (Protected) Mode. De verschillen zijn te zien in tabel 7/4.2-14. Merk op dat sommige registerbits zijn gereserveerd voor toekomstige processoren. Het gebruik van deze bits wordt daarom afgeraden.

I/O ruimte

Geheugen- en input/output

De 486 microprocessor heeft twee aparte fysieke adresruimten: geheugen en I/O (input/output). Gewoonlijk worden perifere schakelingen in de I/O-ruimte geplaatst, hoewel de 486 ook ge“memory-map”te periferieën ondersteunt.

Afmetingen

De I/O-ruimte is 64 kB groot en kan worden verdeeld in 64.000 8 bit poorten, 32.000 16 bit poorten of 16.000 32 bit poorten (of elke combinatie van ten hoogste 64 kB). De 64 kB I/O-adresruimte heeft meer betrekking op fysiek geheugen dan op lineaire adressen, aangezien I/O instructies de segmentatie of paging hardware niet overschrijden.

4.2 80486

Register	Use In Real Mode		Use In Protected Mode		Use In Virtual 8086 Mode	
	Load	Store	Load	Store	Load	Store
General Registers	Yes	Yes	Yes	Yes	Yes	Yes
Segment Register	Yes	Yes	Yes	Yes	Yes	Yes
Flag Register	Yes	Yes	Yes	Yes	IOPL	IOPL*
Control Registers	Yes	Yes	PL = 0	PL = 0	No	Yes
GDTR	Yes	Yes	PL = 0	Yes	No	Yes
IDTR	Yes	Yes	PL = 0	Yes	No	Yes
LDTR	No	No	PL = 0	Yes	No	No
TR	No	No	PL = 0	Yes	No	No
FPU Data Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Control Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Status Registers	Yes	Yes	Yes	Yes	Yes	Yes
FPU Instruction Pointer	Yes	Yes	Yes	Yes	Yes	Yes
FPU Data Pointer	Yes	Yes	Yes	Yes	Yes	Yes
Debug Registers	Yes	Yes	PL = 0	PL = 0	No	No
Test Registers	Yes	Yes	PL = 0	PL = 0	No	No

NOTES:
PL = 0: The registers can be accessed only when the current privilege level is zero.
*IOPL: The PUSHF and POPF instructions are made I/O Privilege Level sensitive in Virtual 86 Mode.

Tabel 7/4.2-14: Verschillen bij het gebruik van de registers.

De M/$\overline{IO}$-pen, die LAAG gaat als een I/O instructie wordt uitgevoerd, werkt als een extra adreslijn, zodat de systeemontwerper eenvoudig kan nagaan welke adresruimte de processor betreedt.

Adresseren

Toegang tot de I/O-poorten ontstaat met de IN en OUT I/O instructies, waarbij het poortadres als een immediate 8 bit constante in de instructie of in het DX-register is opgenomen. De hogere adreslijnen van alle 8 en 16 bit poortadressen zijn gevuld met nullen.

Cache-architectuur

Inleiding

De Enhanced Am486 microprocessoren ondersteunen een superset architectuur van de standaard 486 cache implementatie. Deze opgevoerde architectuur verbetert niet alleen de prestaties van de CPU, maar ook die van het totale systeem.

Write-Through Cache

De standaard 486DX-type write-through cache architectuur wordt gekenmerkt door het volgende:
- Er worden externe lees-toegangen in de cache gezet als die voldoen aan de juiste cache-vereisten.
- Als het adres in de cache tag-array is opgeslagen worden opeenvolgende leesoperaties van data in de cache uitgevoerd.
- Schrijf-operaties naar een geldig adres in de cache worden ge-updated in de cache én in extern geheugen. Deze techniek van data schrijven wordt write-through genoemd.

De write-through cache implementatie maakt dat alle schrijf-operaties naar de externe bus en terug naar het hoofd-geheugen vloeien. Als gevolg daarvan veroorzaakt de write-through cache een grote hoeveelheid verkeer op de externe databus.

Write-back Cache

De write-back architectuur van de microprocessor wordt gekenmerkt door het volgende:
- Externe lees-toegangen worden in de cache gezet als die voldoen aan de juiste cache-vereisten.
- Opeenvolgende lees-operaties van data in de cache worden uitgevoerd als het adres in de cache tag-array is opgeslagen.
- Schrijf-operaties naar een geldig adres in de cache (als die in de write-through toestand staat) worden ge-updated in de cache én in extern geheugen.
- Schrijf-operaties naar een geldig adres in de cache die in de write-back (exclusieve of gemodificeerde) toestand staat, worden alleen ge-updated in de cache.
- Gemodificeerde data wordt naar het externe geheugen teruggeschreven wanneer de gemodificeerde cache-regel door een nieuwe cache-regel wordt vervangen (copy-back operatie) of wanneer een externe bus-master een gemodificeerde cache-regel heeft gestolen (write-back).

Door de write-back cache mogelijkheid wordt het verkeer over de externe bus wel aanzienlijk verminderd, maar het systeem-ontwerp (op het gebied van geheugen-samenhang) wordt er ook moeilijker door. De write-back cache vereist een zwaardere systeem-ondersteuning, omdat de cache data kan bevatten die niet gelijk is aan de data in het

4.2 80486

hoofdgeheugen op hetzelfde adres. Om dit te bereiken is het MESI-protocol (Modified Exclusive Shared and Invalid) van toepassing op de Enhanced Am486 processor. Dit protocol biedt de volgende voordelen:

- Geen onnodig busverkeer. Het protocol identificeert gemeenschappelijke data naar een cache-regel. Deze dynamische identificatie zorgt ervoor dat het verkeer op de externe bus niet meer bedraagt dan wat nodig is voor de samenhang.
- Software-transparant. Aangezien het protocol eruit ziet als een enkel uniform geheugen, is het voor de software niet nodig de samenhang te handhaven of de gemeenschappelijke data te identificeren. Het is zodoende mogelijk dat toepassings-software die is ontwikkeld voor een cache-loos systeem ook nu zonder wijzigingen kan draaien. Er is alleen software-ondersteuning nodig in het bedrijfssysteem om non-cacheable datagebieden te identificeren.

Het MESI-protocol staat toe dat cache-regels in vier toestanden bestaan: gemodificeerd, gemeenschappelijk, exclusief en ongeldig. Daartoe wordt elke tag-entry uitgebreid met twee bits (S1 en S0). Elke tag-entry heeft betrekking op een cache-regel.
In tabel 7/4.2-15 zijn de vier status-mogelijkheden van de cache-regel te zien, terwijl tabel 7/4.2-16 de interpretatie hiervan toont.

S1	S0	Line State
0	0	Invalid
0	1	Exclusive
1	0	Modified
1	1	Shared

Tabel 7/4.2-15: Mogelijke toestanden van een cache-regel.

Situation	Modified	Exclusive	Shared	Invalid
Line valid?	Yes	Yes	Yes	No
External memory is...	out-of-date	valid	valid	status unknown
A write to this cache line...	does not go to the bus	does not go to the bus	goes to the bus and updates	goes directly to the bus

Tabel 7/4.2-16: Interpretatie van de MESI cache-regel status.

Clock-besturing

Inleiding

De Enhanced Am486 CPU wordt aangedreven door een 1x clock die van een phase-locked loop (PLL) gebruik maakt om de twee interne clock-fasen te genereren: fase 1 en fase 2. De stijgende flank van CLK komt overeen met het begin van fase 1. Alle externe timing-parameters zijn gespecificeerd ten opzichte van de stijgende flank van CLK.

Stop Clock

De Enhanced Am486 CPU beschikt ook over een interrupt-mechanisme $\overline{\text{STPCLK}}$, waarmee systeem-hardware het energieverbruik van de CPU kan regelen door het interne clock-signaal naar de CPU te stoppen. De eerste low-power toestand wordt de Stop Grant toestand genoemd. Als het CLK-signaal volledig is gestopt, komt de CPU in de Stop Clock toestand (de toestand met de laagste energie).
Als de CPU een $\overline{\text{STPCLK}}$-interrupt herkent, gebeurt het volgende:

- De processor stopt de executie bij het begin van de volgende instructie (tenzij dit ongedaan wordt gemaakt door een interrupt van hogere prioriteit).
- De processor wacht tot de cache gevuld is.
- De pre-fetch unit wordt gestopt.
- Alle interne pijplijnen en schrijf-buffers worden geleegd.
- Er wordt een Stop Grant cyclus gegenereerd.

4.2 80486

– De interne clock wordt gestopt.
Op dit punt bevindt de CPU zich in de Stop Grant toestand.
De CPU kan niet op een STPCLK-verzoek reageren als hij zich reeds in de HLDA-toestand bevindt, omdat dan de schrijf-buffers niet geleegd kunnen worden en er dus geen Stop Grant cyclus gegenereerd mogelijk is. De stijgende flank van STPCLK geeft aan de CPU het teken dat teruggekeerd kan worden naar het uitvoeren van de instructie die na de geïnterrumpeerde instructie volgt. In tegenstelling tot de normale interrupts (INTR en NMI) heeft STPCLK geen acknowledge-cycli of uitlezen van de interrupt-tabel tot gevolg.

Diverse opties

Externe interrupts in volgorde van prioriteit

In de write-through mode is de prioriteitsvolgorde van externe interrupts:

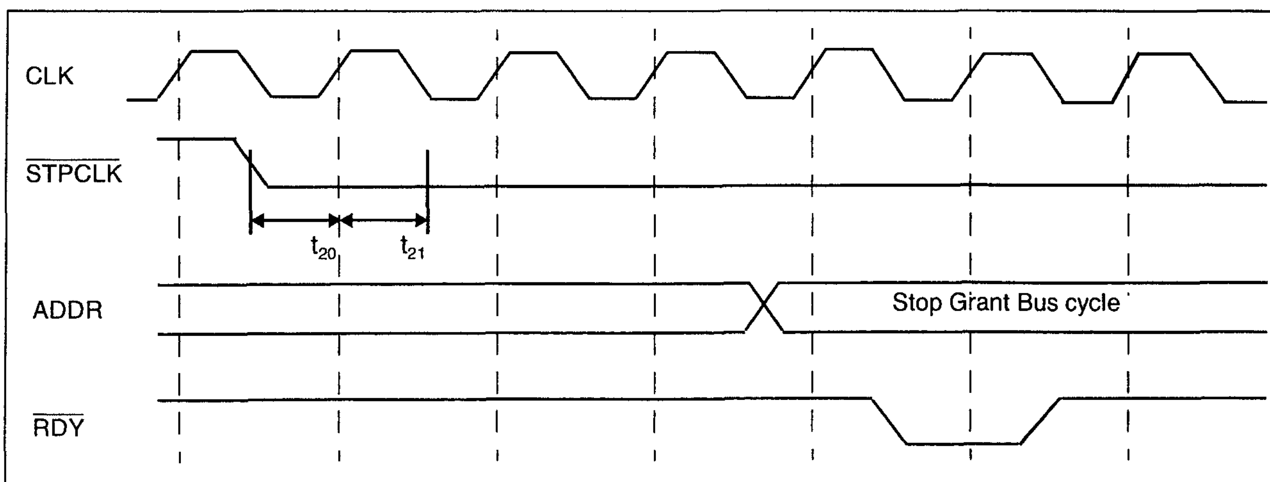
1 RESET/SRESET;
2 FLUSH;
3 SMI;
4 NMI;
5 INTR;
6 STPCLK;

In de write-back mode is de prioriteitsvolgorde van externe interrupts:

1 RESET;
2 FLUSH;
3 SRESET;
4 SMI;
5 NMI;
6 INTR;
7 STPCLK.

Stop Gant buscyclus

De STPCLK-ingang is actief-LAAG en heeft een interne optrek-weerstand. STPCLK is asynchroon, maar er moet worden voldaan aan de setup en hold-tijden om zeker te zijn van herkenning. STPCLK moet actief blijven totdat de Stop Grant buscyclus verschijnt en het systeem reageert met RDY of BRDY.
De tijd die verstrijkt tussen een STPCLK-verzoek en de Stop Grant buscyclus hangt af van de lopende instructie, de hoeveelheid data in de CPU schrijf-buffers en de snelheid van het systeemgeheugen (figuur 7/4.2-20).
Als de CPU in de Stop Grant toestand komt, wordt de interne optrek-weerstand losgekoppeld om de dissipatie te verminderen. Om de Stop Clock toestand te verlaten moet STPCLK HOOG worden gemaakt (niet zwevend).



Figuur 7/4.2-20: Het binnengaan van de Stop Grant toestand.

In tabel 7/4.2-17 is de toestand van de pennen tijdens de Stop Grant Bus-toestanden te zien. De meeste signalen blijven op het niveau waarop zij waren voordat de CPU in de Stop Grant toestand kwam.
De data- en data-pariteits-signalen worden zwevend. Als reactie op het actief worden van HOLD in de Stop Grant toestand (als de CLK-ingang loopt), genereert de CPU HLDA en brengt alle uitgangs- en ingang/uitgangs-signalen in de 3-state toestand. Nadat HOLD is weggevallen komen alle signalen weer terug in de toestand vóór de HOLD/HLDA-volgorde.

Signal	Type	State
A3–A2	O	Previous State
A31–A4	I/O	Previous State
D31–D0	I/O	Floated
BE3–BE0	O	Previous State
DP3–DP0	I/O	Floated
W/R, D/C, M/IO, CACHE	O	Previous State
ADS	O	Inactive
LOCK, PLOCK	O	Inactive
BREQ	O	Previous State
HLDA	O	As per HOLD
BLAST	O	Previous State
FERR	O	Previous State
PCHK	O	Previous State
SMIACT	O	Previous State
HITM	O	Previous State

Tabel 7/4.2-17: Toestand van de pennen tijdens de Stop Grant Bus toestand.

De SRESET-functie

De Enhanced Am486 microprocessorfamilie ondersteunt een soft reset-functie via de SRESET-pen. SRESET maakt dat de processor de executie in een bekende toestand begint. De processor-toestand na SRESET is dezelfde als na RESET, behalve dat de interne caches, CD en NW in CR0, schrijfbuffers, SMBASE registers en floating-point registers de waarden behouden die zij hadden voordat SRESET werd gegeven. De processor begint de executie op fysiek adres FFFFFFF0h. SRESET kan worden gebruikt om de prestaties van DOS-extenders die voor de 286-processor zijn geschreven te verbeteren. Met SRESET kan worden overgeschakeld van Protected naar Real mode met behoud van de interne caches, CR0 en de FPU-toestand. SRESET mag na power-up niet in plaats van RESET worden gebruikt.
In de write-back mode begint de SRESET-volgorde (na detecteren van actieve SRESET) op de volgende instructie-grens, tenzij FLUSH of RESET optreden vóór die grens. Als de SRESET-volgorde eenmaal gestart is, gaat deze door tot het einde, waarna de normale processor-executie begint (of SRESET nu wel of niet aanwezig blijft).

CPU identificatie

De Enhanced Am486 microprocessor ondersteunt twee standaard-methoden om de CPU in een systeem te identificeren. De gevonden waarden worden dynamisch toegekend, gebaseerd op het CPU-type (DX2 of DX4) en de status van de WB/WT-pen (LAAG = write-through; HOOG = write-back) bij RESET.
Het DX-register bevat aan het einde van RESET altijd een identifier. Het hoogste byte van DX (DH) bevat 04 en het laagste byte (DL) een CPU-type/stepping identifier (zie tabel 7/4.2-18).
De Enhanced Am486 microprocessor- familie implementeert een nieuwe instructie die de software informeert over de familie en het model van de microprocessor waarop hij werkt. De ondersteuning van deze instructie wordt aangegeven door de aanwezigheid van een modificeerbaar bit in positie EFLAGS.21 (ook wel het EFLAGS-ID-bit genoemd). Bij het resetten (met RESET of SRESET) wordt dit bit nul gemaakt om compatibel te zijn met bestaande processor-ontwerpen.

Instructieset

De 486 instructieset bestaat uit 11 categorieën operaties:

- Data transfer;
- Arithmetic;
- Shift/Rotate;
- String Manipulation;
- Bit Manipulation;
- Control Transfer;
- High Level Language Support;
- Operating System Support;
- Processor Control;
- Floating Point;
- Floating Point Control.

Alle 486 instructies werken met 0, 1, 2 of 3 operands. De operand kan zich daarbij in een register, in de instructie zelf of in geheugen bevinden. De meeste nul-operand instructies, zoals CLI en STI hebben slechts één byte nodig. Eén-operand instructies zijn gewoonlijk twee byte lang. De gemiddelde instructie heeft een lengte van 3,2 byte. Aangezien de 486 een 32 byte instructie-wachtrij (queue) heeft, kunnen gemiddeld 10 instructies van te voren worden opgehaald.
Door twee operands te gebruiken kunnen de volgende, veel gebruikte instructies worden uitgevoerd:

- register-naar-register;
- geheugen-naar-register;
- geheugen-naar-geheugen;
- immediate-naar-register;
- register-naar-geheugen;
- immediate-naar-geheugen.

De operands kunnen 8, 16 of 32 bit lang zijn. Over het algemeen geldt dat wanneer 32 bit code wordt uitgevoerd die voor de 486 of 386 is geschreven, de operands 8 of 32 bit zijn. Bij het uitvoeren van bestaande 16 bit 286 of 86 code zijn de operands 8 of 16 bit. Aan alle instructies kan een prefix worden toegevoegd waarmee de initiële lengte van de operands wordt omzeild.

Data-typen

Inleiding

De 486 microprocessor kan met veel verschillende data-typen werken (figuur 7/4.2-21). In de volgende beschrijvingen bestaat de FPU uit de floating point registers en de CPU uit de basisarchitectuur-registers.

Unsigned Data

Data-typen zonder teken (unsigned data) worden niet ondersteund door de FPU. In een byte is het minst belangrijke bit (LSB) bit 0, het belangrijkste (MSB) is bit 7. Er wordt gewerkt met de volgende unsigned datatypen:

- Byte: 8 bit;
- Word: 16 bit;
- Dword: 32 bit;

(alle zonder teken).

Signed Data

Alle data-typen met teken (signed data) worden in 2's complement notatie gebruikt. Deze data-typen bevatten twee velden: een teken en een grootheid. Het tekenbit is het belangrijkste bit (MSB).
Het getal is negatief als het tekenbit = 1 is en positief als dit = 0 is. Het veld van de grootheid bestaat uit de resterende bits in het getal. Er wordt gewerkt met de volgende signed data-typen:

- 8-bit integer: +/-8 bit;
- 16-bit integer: +/-16 bit;
- 32-bit integer: +/-32 bit;
- 64-bit integer: +/-64 bit;

(alle inclusief teken).
De FPU ondersteunt alleen 16, 32 en 64 bit integers en de CPU alleen 8, 16 en 32 bit integers.

CPU Type and Cache Mode	Component ID (DH)	Revision ID (DL)
DX2 in write-through mode	04	3x
DX2 in write-back mode	04	7x
DX4 in write-through mode	04	8x
DX4 in write-back mode	04	9x

Tabel 7/4.2-18: CPU ID-codes.

4.2 80486

Data Format	Supported by Base Registers	Supported by FPU	Range	Precision
Byte	X		0–255	8 bits
Word	X		0–64K	16 bits
Dword	X		0–4G	32 bits
8-Bit Integer	X		10^2	8 bits
16-Bit Integer	X	X	10^4	16 bits
32-Bit Integer	X	X	10^9	32 bits
64-Bit Integer		X	10^{19}	64 bits
8-Bit Unpacked BCD	X		0–9	1 Digit
8-Bit Packed BCD	X		0–9	2 Digits
80-Bit Packed BCD		X	$\pm 10^{\pm 18}$	18 Digits
Single Precision Real		X	$\pm 10^{\pm 38}$	24 Bits
Double Precision Real		X	$\pm 10^{\pm 308}$	53 Bits
Extended Precision Real		X	$\pm 10^{\pm 4932}$	64 Bits

Figuur 7/4.2-21: Data-typen in de 486 microprocessor.

4.2 80486

Floating Point Data

Data-typen met zwevende komma (floating point data) bevatten bij de 486 processor drie velden: teken, significand en exponent. Het tekenveld is één bit: het MSB van het floating point getal. Een getal is negatief als het tekenbit = 1 is. De significand bestaat uit de significante bits van het getal en het exponentveld bevat de macht van 2 die nodig is om de significand in te schalen. Floating point data wordt alleen door de FPU ondersteund.

- Single precision real:
  23 bit significand en 8 bit exponent
  (= totaal 32 bits).
- Double precision real:
  52 bit significand en 11 bit exponent
  (= totaal 64 bits).
- Extended precision real:
  64 bit significand en 15 bit exponent
  (= totaal 80 bits).

BCD data

De 486 ondersteunt ook gepakte en ongepakte binair gecodeerde decimale (BCD) data-typen. Een gepakt BCD data-type bevat twee cijfers (digits) per byte. Het laagste cijfer wordt daarbij bepaald door de bits 0 tot en met 3 en het hoogste cijfer door de bits 4 tot en met 7. Een ongepakt BCD data-type bevat één cijfer per byte dat is opgeslagen in de bits 0 tot en met 3.

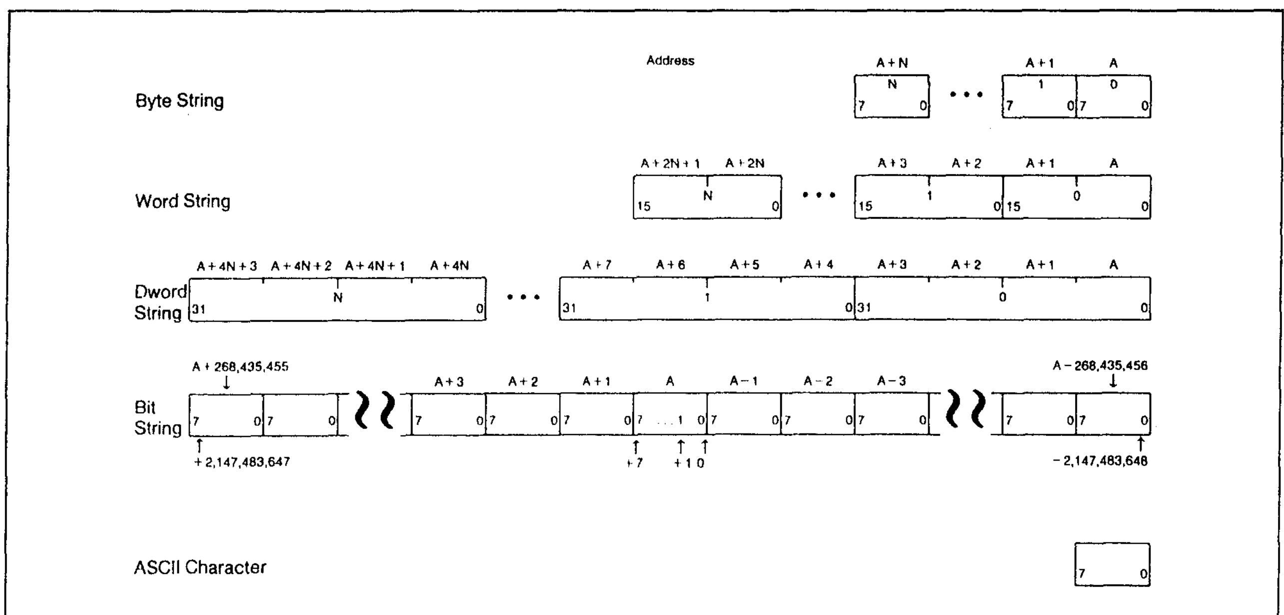
De CPU ondersteunt 8 bit gepakte en ongepakte BCD data-typen; de FPU ondersteunt slechts 80 bit gepakte BCD data-typen.

String data

Een string data-type is een aaneengesloten reeks bits, bytes, words of dwords. Een string kan tussen 1 byte en 4 GB groot zijn (zie figuur 7/4.2-22). String data-typen worden alleen ondersteund door de CPU. Er wordt gewerkt met de volgende strings:

- Byte string:
  aaneengesloten volgorde van bytes.
- Word string:
  aaneengesloten volgorde van words.
- Dword string:
  aaneengesloten volgorde van dwords.
- Bit string:
  een set van aaneengesloten bits.

In de 486 microprocessor kunnen strings maximaal 4 GB lang zijn.



Figuur 7/4.2-22: String en ASCII data typen.

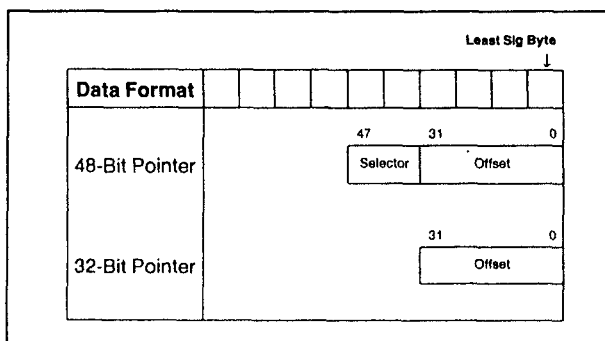
ASCII data

De 486 ondersteunt natuurlijk ook ASCII strings (American Standard Code for Information Interchange) en kan er rekenkundige bewerkingen mee doen, zoals optellen en delen. Alleen de CPU kan met ASCII data (figuur 7/4.2-22) werken.

Pointer data

Een pointer data-type bestaat uit een waarde die het adres van een stuk data bevat. De 486 ondersteunt twee typen pointers (zie figuur 7/4.2-23):

- 48 bit pointer:
  16 bit selector en 32 bit offset,
- 32 bit pointer:
  32 bit offset.



Figuur 7/4.2-23: Pointer data typen.

Elektrische karakteristieken

Tenslotte worden de elektrische- en timing-karakteristieken van de verschillende Enhanced Am486 processoren vermeld, die werken met busclock-frequenties van 25, 33 en 40 MHz. De schakeltijden in de tabellen 7/4.2-21, -22 en -23 bestaan uit vertragingen van de uitgangssignalen en set-up en hold-tijden. Al deze tijden zijn relatief aan de stijgende flank van het CLK-signaal. De in deze tabellen genoemde figuren 39 t/m 44 komen overeen met de figuren 7/4.2-24 t/m -29. Evenzo zijn de bij de Boundary Scan Test behorende figuren 45 en 46 in dit deel 7/4.2-30 en -31 genoemd.

ABSOLUTE MAXIMUM RATINGS

Case Temperature under Bias ...	− 65°C to +110°C
Storage Temperature	− 65°C to +150°C
Voltage on any pin with respect to ground	− 0.5 V to V_{cc} +2.6 V
Supply voltage with respect to V_{SS}	− 0.5 V to +4.6 V

OPERATING RANGES

Commercial (C) Devices

T_{CASE}	0°C to 85°C
V_{CC}	3.3 V ±0.3 V

Tabel 7/4.2-19: Maximaal toegelaten waarden en bedrijfscondities.

4.2 80486

V_{CC} = 3.3 V ± 0.3 V; T_{CASE} = 0°C to + 85°C

Symbol	Parameter	Min	Max	Notes
V_{IL}	Input Low Voltage	– 0.3 V	+0.8 V	
V_{IH}	Input High Voltage	2.0 V	V_{CC} + 2.4 V	
V_{OL}	Output Low Voltage		0.45 V	Note 1
V_{OH}	Output High Voltage	2.4 V		Note 2
I_{CC}	Power Supply Current: 66 MHz 75 MHz 80 MHz 100 MHz 120 MHz		660 mA 750 mA 800 mA 1000 mA 1200 mA	Typical supply current: 528 mA @ 66 MHz, 600 mA @ 75 MHz, 640 mA @ 80 MHz, 800 mA @ 100 MHz, and 960 mA @ 120 MHz Inputs at rails, outputs unloaded.
$I_{CCSTOPGRANT}$ or $I_{CCAUTOHALT}$	Input Current in Stop Grant or Auto Halt Mode: 66 MHz 75 MHz 80 MHz 100 MHz 120 MHz		66 mA 75 mA 80 mA 100 mA 120 mA	Typical supply current for Stop Grant or Auto Halt Mode: 20 mA @ 66 MHz and 75 MHz, 30 mA @ 80 MHz, 50 mA @ 100 MHz, and 60 mA @ 120 MHz.
$I_{CCSTPCLK}$	Input Current in Stop Clock Mode		5 mA	Typical supply current in Stop Clock Mode is 600 µA.
I_{LI}	Input Leakage Current		± 15 µA	Note 3
I_{IH}	Input Leakage Current		200 µA	Note 4
I_{IL}	Input Leakage Current		– 400 µA	Note 5
I_{LO}	Output Leakage Current		± 15 µA	
C_{IN}	Input Capacitance		10 pF	F_C = 1 MHz (Note 6)
C_O	I/O or Output Capacitance		14 pF	F_C = 1 MHz (Note 6)
C_{CLK}	CLK Capacitance		12 pF	F_C = 1 MHz (Note 6)

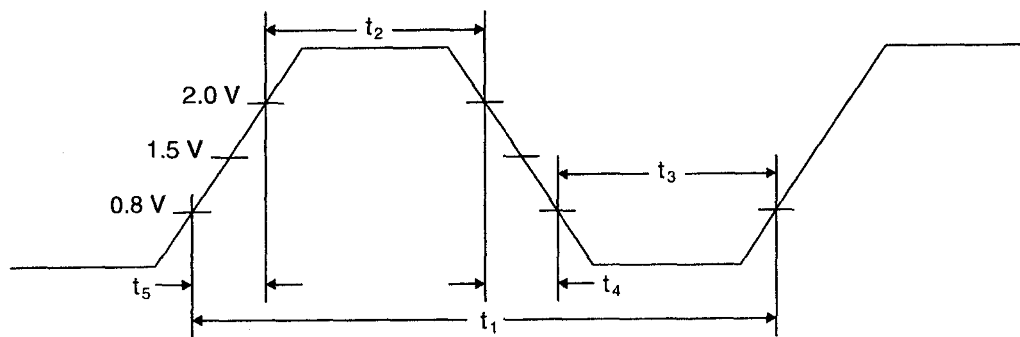
Notes:

1. *This parameter is measured at: Address, Data, $\overline{BE}n$ = 4.0 mA; Definition, Control = 5.0 mA*
2. *This parameter is measured at: Address, Data, $\overline{BE}n$ = - 1.0 mA; Definition, Control = - 0.9 mA*
3. *This parameter is for inputs without internal pull-ups or pull-downs and $0 \leq V_{IN} \leq V_{CC}$.*
4. *This parameter is for inputs with internal pull-downs and V_{IH} = 2.4 V.*
5. *This parameter is for inputs with internal pull-ups and V_{IL} = 0.45 V.*
6. *Not 100% tested.*

Tabel 7/4.2-20: Gelijkspannings-karakteristieken van de Enhanced Am486-typen.

4.2 80486

Waveform	Inputs	Outputs
	Must be steady	Will be steady
	May change from H to L	Will change from H to L
	May change from L to H	Will change from L to H
	Don't care; any change permitted	Changing; state unknown
	Does not apply	Center line is High-impedance "Off" state



Figuur 7/4.2-24: (39) CLK golfvormen en betekenis van de bij de overige golfvormen gebruikte aanduidingen.

4.2 80486

V_{CC} = 3.3 V ±0.3 V; T_{CASE} = 0°C to + 85°C; C_L = 50 pF unless otherwise specified

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	33	MHz		Note 2
t_1	CLK Period	30	125	ns	39	
t_{1a}	CLK Period Stability		0.1%	Δ		Adjacent Clocks Notes 3 and 4
t_2	CLK High Time at 2 V	11		ns	39	Note 3
t_3	CLK Low Time at 0.8 V	11		ns	39	Note 3
t_4	CLK Fall Time (2 V–0.8 V)		3	ns	39	Note 3
t_5	CLK Rise Time (0.8 V–2 V)		3	ns	39	Note 3
t_6	A31–A2, PWT, PCD, BE3–BE0, M/IO, D/C, CACHE, W/R, ADS, LOCK, FERR, BREQ, HLDA, SMIACT, HITM Valid Delay	3	14	ns	40	Note 5
t_7	A31–A2, PWT, PCD, BE3–BE0, M/IO, D/C, CACHE, W/R, ADS, LOCK Float Delay	3	20	ns	41	Note 3
t_8	PCHK Valid Delay	3	14	ns	42	
t_{8a}	BLAST, PLOCK, Valid Delay	3	14	ns	40	
t_9	BLAST, PLOCK, Float Delay	3	20	ns	41	Note 3
t_{10}	D31–D0, DP3–DP0 Write Data Valid Delay	3	14	ns	40	
t_{11}	D31–D0, DP3–DP0 Write Data Float Delay	3	20	ns	41	Note 3
t_{12}	EADS, INV, WB/WT Setup Time	5		ns	43	
t_{13}	EADS, INV, WB/WT Hold Time	3		ns	43	
t_{14}	KEN, BS16, BS8 Setup Time	5		ns	43	
t_{15}	KEN, BS16, BS8 Hold Time	3		ns	43	
t_{16}	RDY, BRDY Setup Time	5		ns	44	
t_{17}	RDY, BRDY Hold Time	3		ns	44	
t_{18}	HOLD, AHOLD Setup Time	6		ns	43	
t_{18a}	BOFF Setup Time	7		ns	43	
t_{19}	HOLD, AHOLD, BOFF Hold Time	3		ns	43	
t_{20}	RESET, FLUSH, A20M, NMI, INTR, IGNNE, STPCLK, SRESET, SMI Setup Time	5		ns	43	Note 5
t_{21}	RESET, FLUSH, A20M, NMI, INTR, IGNNE, STPCLK, SRESET, SMI Hold Time	3		ns	43	Note 5
t_{22}	D31–D0, DP3–DP0, A31–A4 Read Setup Time	5		ns	43, 44	
t_{23}	D32–D0, DP3–DP0, A31–A4 Read Hold Time	3		ns	43, 44	

Notes:
1. *Specifications assume C_L = 50 pF. I/O Buffer model must be used to determine delays due to loading (trace and component). First Order I/O buffer models for the processor are available.*
2. *0 MHz operation guaranteed during stop clock operation.*
3. *Not 100% tested. Guaranteed by design characterization.*
4. *For faster transitions (>0.1% between adjacent clocks), use the Stop Clock protocol to switch operating frequency.*
5. *All timings are referenced at 1.5 V (as illustrated in the listed figures) unless otherwise noted.*

Tabel 7/4.2-21: Schakeltijden voor Enhanced Am486-typen met een bus-frequentie van 33 MHz (66 MHz of 100 MHz-type). Zie ook de figuren 7/4.2-24 tot en met -29.

4.2 80486

V_{CC} = 3.3 V ± 0.3 V; T_{CASE} = 0°C to + 85°C; C_L = see Note 1

Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	40	MHz		Note 2
t_1	CLK Period	25	125	ns	39	
t_{1a}	CLK Period Stability		0.1%	Δ		Adjacent Clocks Notes 3 and 4
t_2	CLK High Time at 2 V	9		ns	39	Note 3
t_3	CLK Low Time at 0.8 V	9		ns	39	Note 3
t_4	CLK Fall Time (2 V–0.8 V)		3	ns	39	Note 3
t_5	CLK Rise Time (0.8 V–2 V)		3	ns	39	Note 3
t_6	A31–A2, PWT, PCD, $\overline{BE3}$-$\overline{BE0}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{CACHE}$, W/$\overline{R}$, $\overline{ADS}$, $\overline{LOCK}$, $\overline{FERR}$, BREQ, HLDA, $\overline{SMIACT}$, $\overline{HITM}$ Valid Delay	3	14	ns	40	Note 5
t_7	A31–A2, PWT, PCD, $\overline{BE3}$-$\overline{BE0}$, M/$\overline{IO}$, D/$\overline{C}$, $\overline{CACHE}$, W/$\overline{R}$, $\overline{ADS}$, $\overline{LOCK}$ Float Delay	3	18	ns	41	Note 3
t_8	$\overline{PCHK}$ Valid Delay	3	16	ns	42	
t_{8a}	$\overline{BLAST}$, $\overline{PLOCK}$, Valid Delay	3	18	ns	40	
t_9	$\overline{BLAST}$, $\overline{PLOCK}$, Float Delay	3	16	ns	41	Note 3
t_{10}	D31–D0, DP3–DP0 Write Data Valid Delay	3	16	ns	40	
t_{11}	D31–D0, DP3–DP0 Write Data Float Delay	3	18	ns	41	Note 3
t_{12}	$\overline{EADS}$, INV, WB/$\overline{WT}$ Setup Time	5		ns	43	
t_{13}	$\overline{EADS}$, INV, WB/$\overline{WT}$ Hold Time	3		ns	43	
t_{14}	$\overline{KEN}$, $\overline{BS16}$, $\overline{BS8}$ Setup Time	5		ns	43	
t_{15}	$\overline{KEN}$, $\overline{BS16}$, $\overline{BS8}$ Hold Time	3		ns	43	
t_{16}	$\overline{RDY}$, $\overline{BRDY}$ Setup Time	5		ns	44	
t_{17}	$\overline{RDY}$, $\overline{BRDY}$ Hold Time	3		ns	44	
t_{18}	HOLD, AHOLD Setup Time	6		ns	43	
t_{18a}	$\overline{BOFF}$ Setup Time	8		ns	43	
t_{19}	HOLD, AHOLD, $\overline{BOFF}$ Hold Time	3		ns	43	
t_{20}	RESET, $\overline{FLUSH}$, $\overline{A20M}$, NMI, INTR, $\overline{IGNNE}$, $\overline{STPCLK}$, SRESET, $\overline{SMI}$ Setup Time	5		ns	43	Note 5
t_{21}	RESET, $\overline{FLUSH}$, $\overline{A20M}$, NMI, INTR, $\overline{IGNNE}$, $\overline{STPCLK}$, SRESET, $\overline{SMI}$ Hold Time	3		ns	43	Note 5
t_{22}	D31–D0, DP3–DP0, A31–A4 Read Setup Time	5		ns	43, 44	
t_{23}	D32–D0, DP3–DP0, A31–A4 Read Hold Time	3		ns	43, 44	

Tabel 7/4.2-22: Schakeltijden voor Enhanced Am486-typen met een bus-frequentie van 40 MHz (80 MHz of 120 MHz-type). Notes: zie onderaan tabel 7/4.2-21.

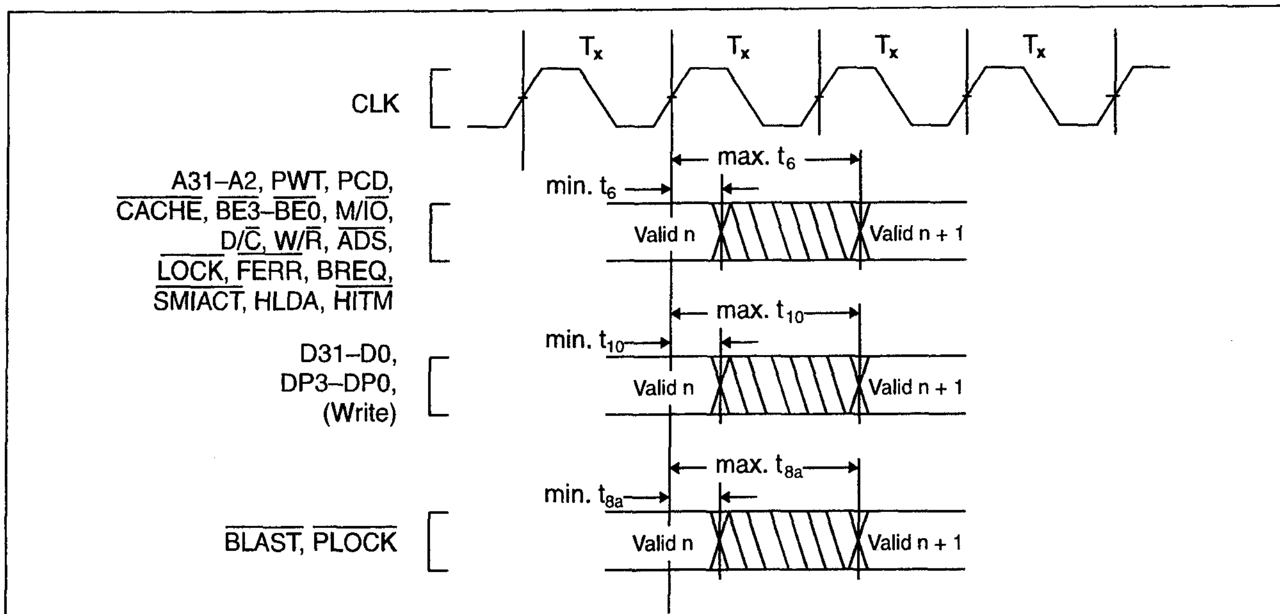
4.2 80486

V_{CC} = 3.3 V ± 0.3 V; T_{CASE} = 0°C to + 85°C; C_L = see Note 1

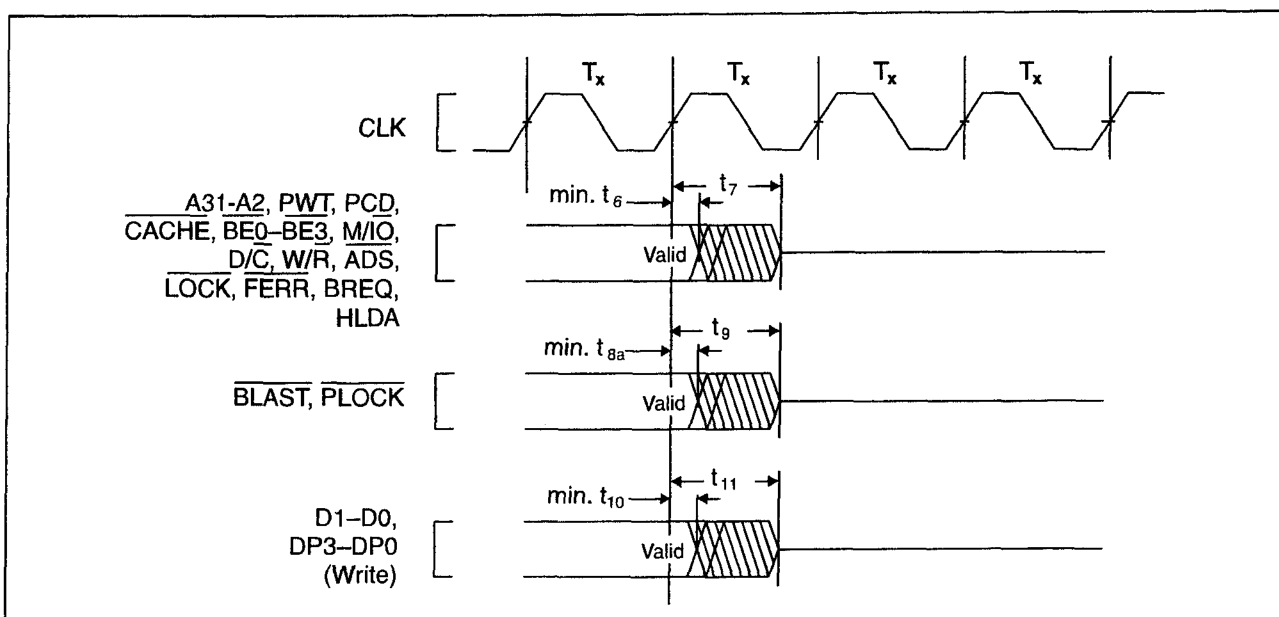
Symbol	Parameter	Min	Max	Unit	Figure	Notes
	Frequency	8	25	MHz		Note 2
t_1	CLK Period	40	125	ns	39	
t_{1a}	CLK Period Stability		0.1%	Δ		Adjacent Clocks Notes 3 and 4)
t_2	CLK High Time at 2 V	14		ns	39	Note 3
t_3	CLK Low Time at 0.8 V	14		ns	39	Note 3
t_4	CLK Fall Time (2 V–0.8 V)		4	ns	39	Note 3
t_5	CLK Rise Time (0.8 V–2 V)		4	ns	39	Note 5
t_6	A31–A2, PWT, PCD, BE3–BE0, M/IO D/C, CACHE, W/R, ADS, LOCK, FERR, BREQ, HLDA, SMIACT, HITM Valid Delay	3	19	ns	40	Note 4
t_7	A31–A2, PWT, PCD, BE3–BE0, M/IO, D/C, CACHE, W/R, ADS, LOCK Float Delay	3	28	ns	41	Note 3
t_8	PCHK Valid Delay	3	24	ns	42	
t_{8a}	BLAST, PLOCK, Valid Delay	3	24	ns	40	
t_9	BLAST, PLOCK, Float Delay	3	28	ns	41	Note 3
t_{10}	D31–D0, DP3–DP0 Write Data Valid Delay	3	20	ns	40	
t_{11}	D31–D0, DP3–DP0 Write Data Float Delay	3	28	ns	41	Note 3
t_{12}	EADS, INV, WB/WT Setup Time	8		ns	43	
t_{13}	EADS, INV, WB/WT Hold Time	3		ns	43	
t_{14}	KEN, BS16, BS8 Setup Time	8		ns	43	
t_{15}	KEN, BS16, BS8 Hold Time	3		ns	43	
t_{16}	RDY, BRDY Setup Time	8		ns	44	
t_{17}	RDY, BRDY Hold Time	3		ns	44	
t_{18}	HOLD, AHOLD Setup Time	8		ns	43	
t_{18a}	BOFF Setup Time	8		ns	43	
t_{19}	HOLD, AHOLD, BOFF Hold Time	3		ns	43	
t_{20}	RESET, FLUSH, A20M, NMI, INTR, IGNNE, STPCLK, SRESET, SMI Setup Time	8		ns	43	Note 5
t_{21}	RESET, FLUSH, A20M, NMI, INTR, IGNNE, STPCLK, SRESET, SMI Hold Time	3		ns	43	Note 5
t_{22}	D31–D0, DP3–DP0, A31–A4 Read Setup Time	5		ns	43, 44	
t_{23}	D32–D0, DP3–DP0, A31–A4 Read Hold Time	3		ns	43, 44	

Tabel 7/4.2-23: Schakeltijden voor het 75 MHz Enhanced Am486-type (bus-frequentie: 25 MHz). Notes: zie onderaan tabel 7/4.2-21.

4.2 80486

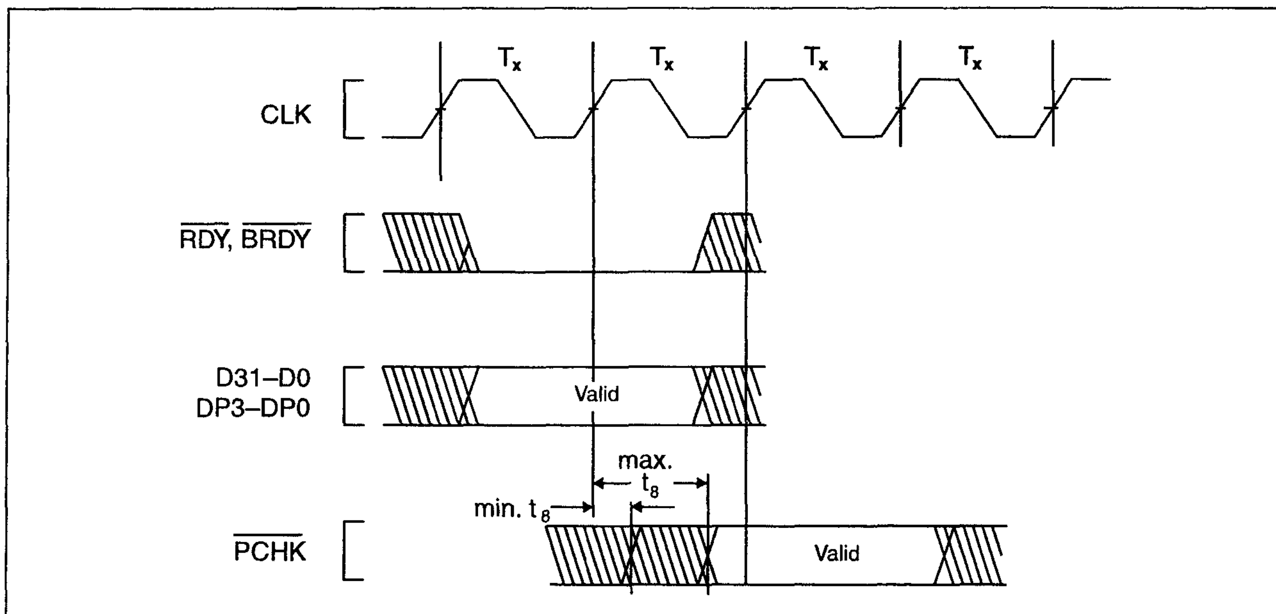


Figuur 7/4.2-25: (40) Vertragingstijden tot het geldig worden van de uitgangssignalen.

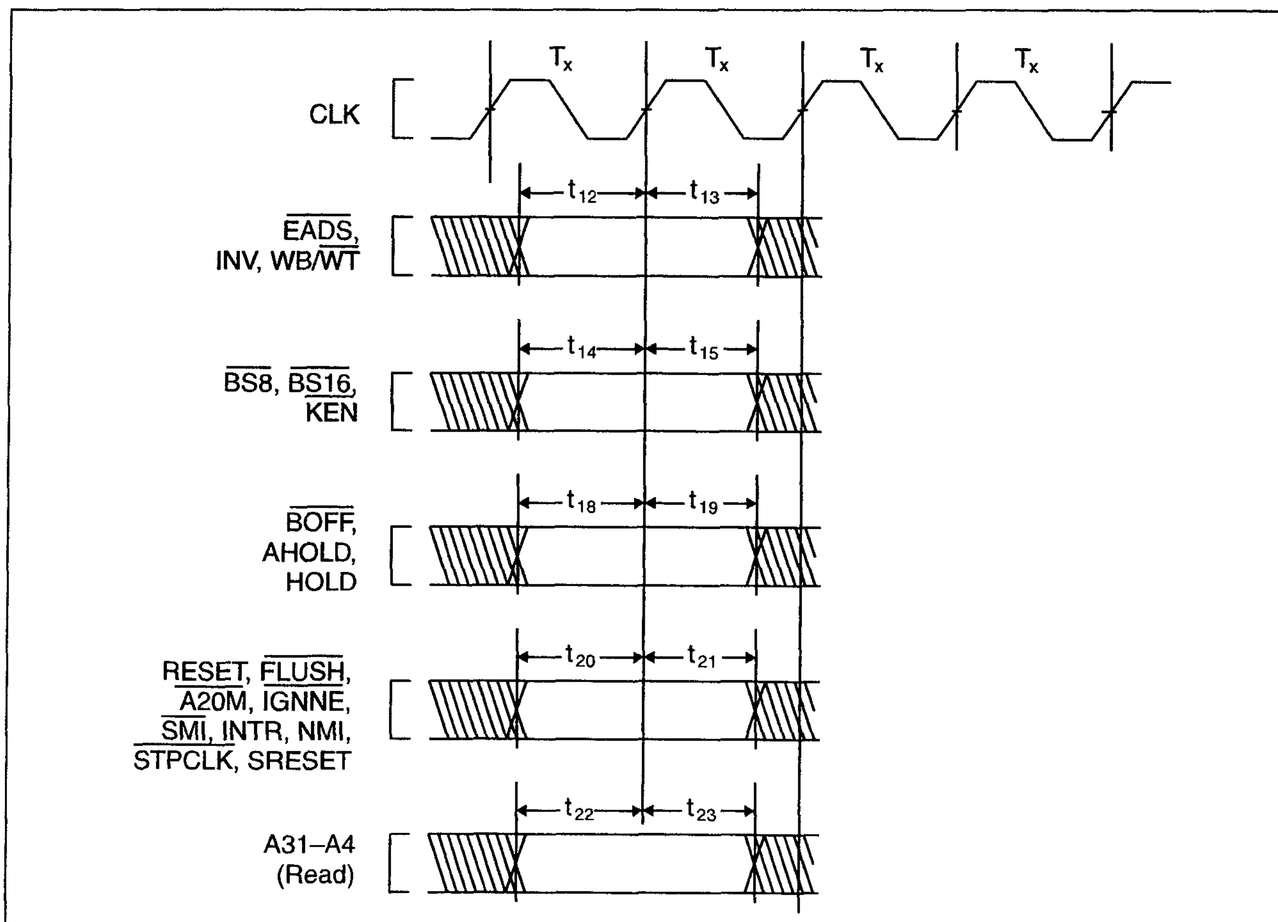


Figuur 7/4.2-26: (41) Vertragingstijden tot het zwevend worden van de uitgangssignalen.

4.2 80486

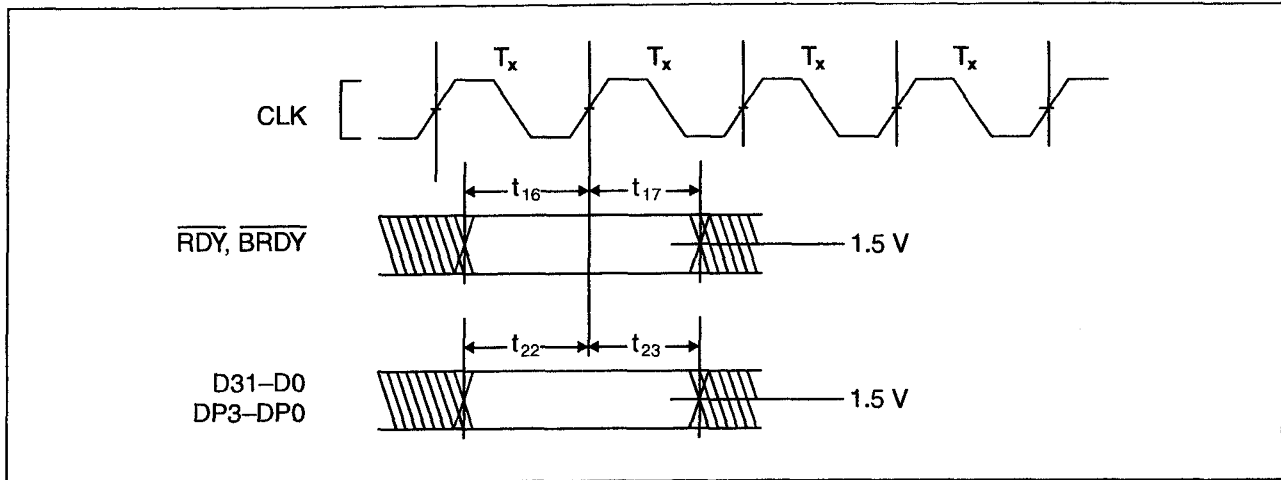


Figuur 7/4.2-27: (42) Vertragingstijd tot het geldig worden van PCHK.



Figuur 7/4.2-28: (43) Input Setup en Hold timing.

4.2 80486



Figuur 7/4.2-29: (44) Input Setup en Hold timing van RDY en BRDY.

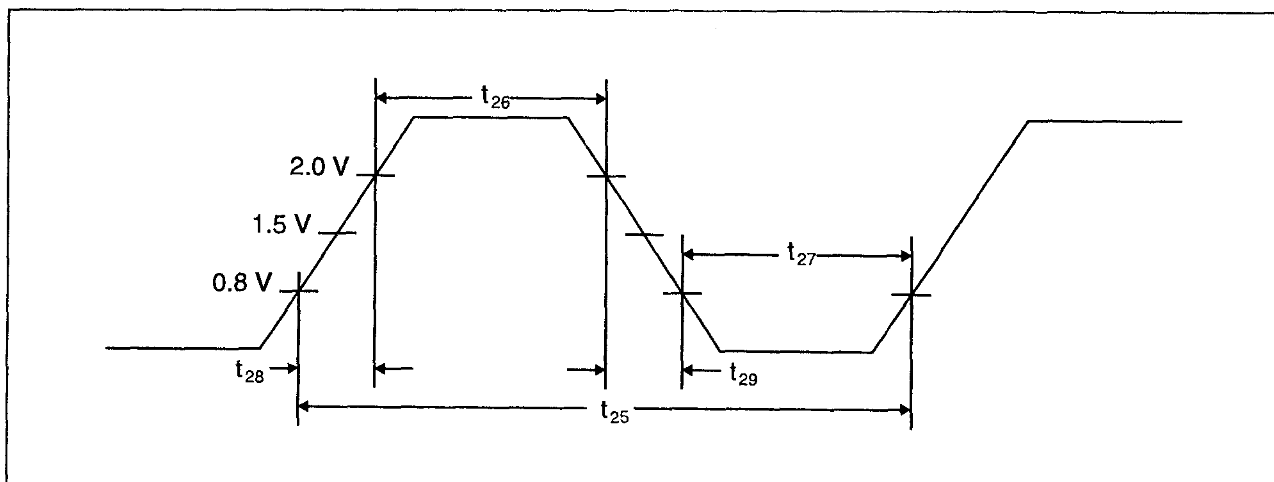
V_{CC} = 3.3 V ±0.3 V; T_{CASE} = 0°C to +85°C; C_L = 50 pF unless otherwise specified

Symbol	Parameter	Min	Max	Unit	Figure	Notes
t_{24}	TCK Frequency		25	MHz		1X Clock
t_{25}	TCK Period	40		ns	45, 46	Note 1
t_{26}	TCK High Time at 2 V	10		ns	45	
t_{27}	TCK Low Time at 0.8 V	10		ns	45	
t_{28}	TCK Rise Time (0.8 V–2 V)		4	ns	45	Note 2
t_{29}	TCK Fall Time (2 V–0.8 V)		4	ns	45	Note 2
t_{30}	TDI, TMS Setup Time	8		ns	46	Note 3
t_{31}	TDI, TMS Hold Time	7		ns	46	Note 3
t_{32}	TDO Valid Delay	3	25	ns	46	Note 3
t_{33}	TDO Float Delay		36	ns	46	Note 3
t_{34}	All Outputs (Non-Test) Valid Delay	3	25	ns	46	Note 3
t_{35}	All Outputs (Non-Test) Float Delay		30	ns	46	Note 3
t_{36}	All Inputs (Non-Test) Setup Delay	8		ns	46	Note 3
t_{37}	All inputs (Non-Test) Hold Time	7		ns	46	Note 3

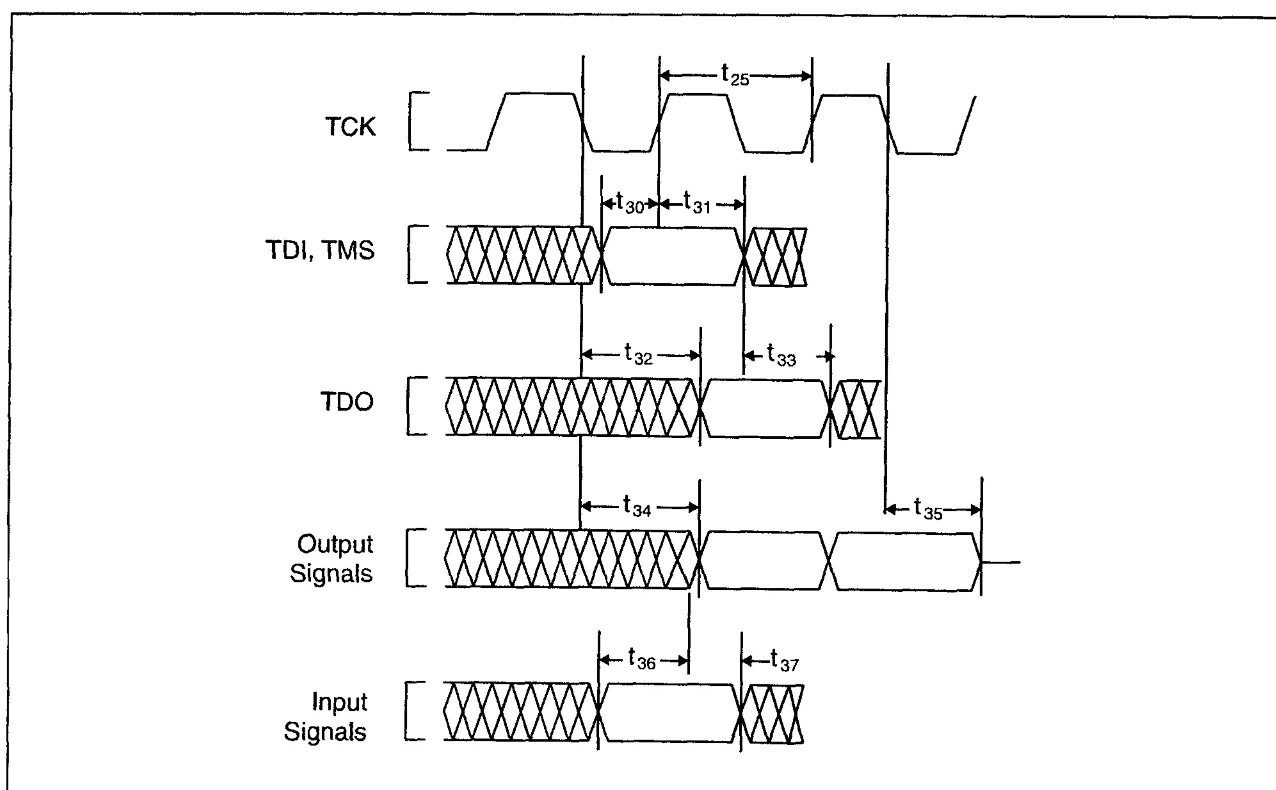
Notes:
1. *TCK period ≥ CLK period.*
2. *Rise/Fall times can be relaxed by 1 ns per 10-ns increase in TCK period.*
3. *Parameter measured from TCK.*

Tabel 7/4.2-24: Schakeltijden voor Boundary Scan Test-signalen bij 25 MHz. Zie ook de figuren 45 (7/4.2-30) en 46 (7/4.2-31).

4.2 80486



Figuur 7/4.2-30: (45) TCK golfvorm.



Figuur 7/4.2-31: (46) Timing van de Boundary Scan Test-signalen.

4.2 80486

7/6

Microcontrollers

Inhoud

7/6.1

8-bit microcontrollers van de 8048-familie

Inleiding

Familie-overzicht

De microcontrollers uit de MCS-48 (Intel) of TLCS-48 (Toshiba) serie hebben alle dezelfde CPU als het type 8048, terwijl ook de instructieset en de aansluitingen voor de hele familie dezelfde zijn. Ze verschillen van elkaar door het al dan niet aanwezig zijn van interne ROM of EPROM en de grootte van de interne RAM. In tabel 7/6.1-1 zijn de leden met hun belangrijkste kenmerken samengevat.

type	RAM	ROM	I/O-lijnen
8048	64 x 8	1K x 8 ROM	27
8049	128 x 8	2K x 8 ROM	27
8050	256 x 8	4K x 8 ROM	27
8035	64 x 8	geen	27
8039	128 x 8	geen	27
8040	256 x 8	geen	27
8748	64 x 8	1K x 8 EPROM	27
8749	128 x 8	2K x 8 EPROM	27

Tabel 7/6.1-1: Leverbare typen uit de 8048-serie.

In dit gedeelte worden de gemeenschappelijke eigenschappen van de gehele familie behandeld, waarna de specifieke timing- en elektrische kenmerken van de "gewone" (met en zonder ROM) en de EPROM-houdende typen apart worden beschreven.

Leveranciers

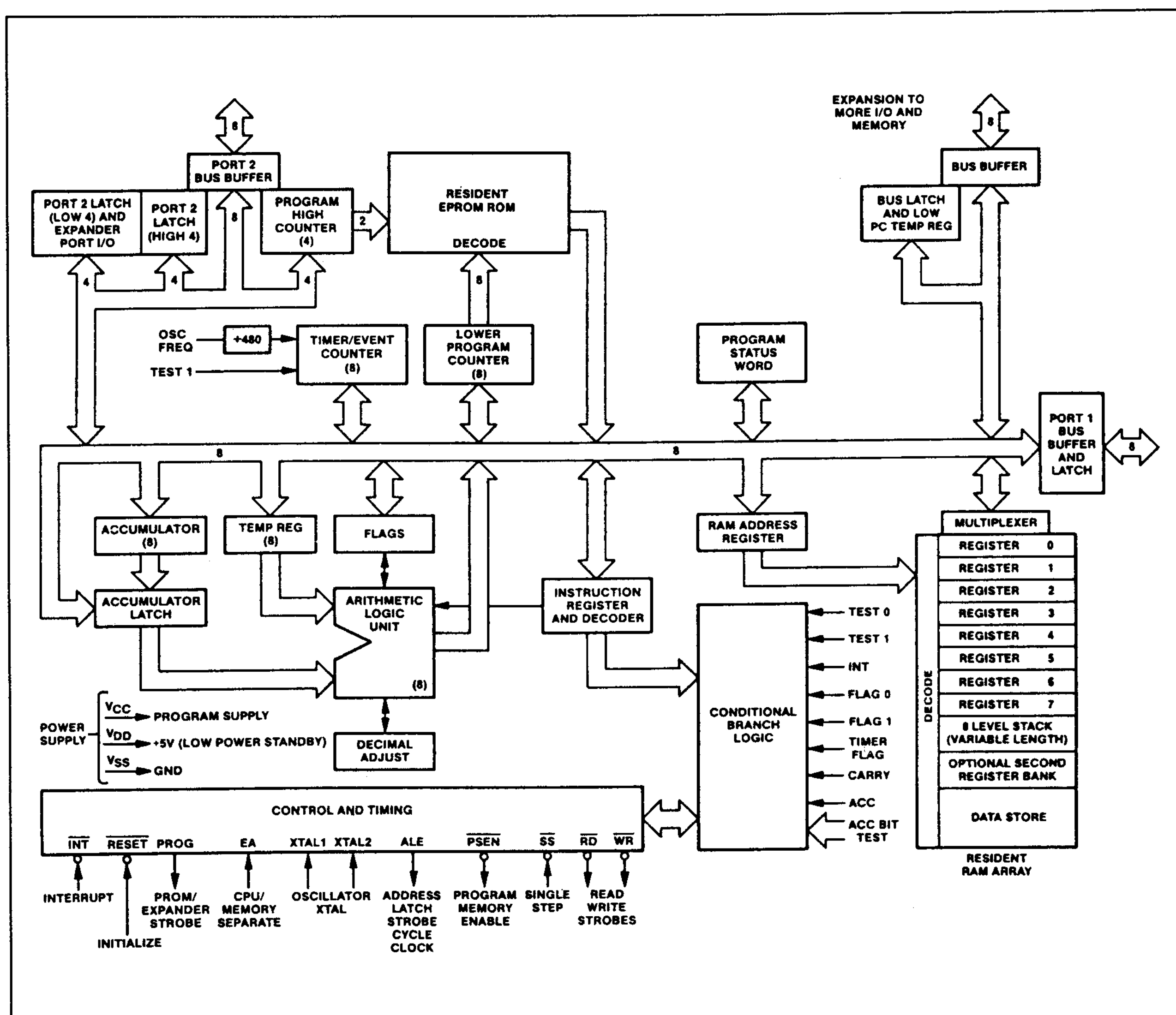
- Fujitsu
  MBL 8039N
  MBL 8048N/E/H, 8049N/E/H
  MBL 8749N/H, 8649N/H (OTP-versie van 8749)
  MBL 80C39N/H, 80C49N/H
- Intel
  8035AHL, 8039AHL, 8040AHL
  8048AH, 8049AH, 8050AH
  8748H, 8749H
- NEC
  µPD 8035LC, µPD 8039LC
  µPD 8048C/D, 8049C/D
  µPD 8748D(-8)
  µPD 80C35C/D, 80C39C/D
  µPD 80C48C/D, 80C49C/D
- OKI
  MSM 80C35, 80C39, 80C40
  MSM 80C48, 80C49, 80C50
- Philips
  MAB 8035HL, 8039HL, 8040HL
  MAB 8048H, 8049H, 8050H
  PCB 80C39, 80C49
- Siemens
  SAB 8035L-P, 8048P
- Toshiba
  TMP 8035P, 8039P/AP/P-6
  TMP 8048P, 8049P/AP/P-6
  TMP 80C35AP, 80C39AP, 80C40AP
  TMP 80C48AP/AF, 80C49AP/AF, 80C50AP/AF

Architectuur

Inleiding

In het volgende worden de delen waaruit de controller is opgebouwd apart behandeld.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-1: Blokschema van de 8048/8748/8049/8749/8050 microcontrollers.

Hierbij wordt uitgegaan van de 8048 die representatief is voor de hele familie (zie het blokschema, figuur 7/6.1-1).

Rekenkundig gedeelte

Het rekenkundige (aritmetische) deel van de processor voert de belangrijkste datamanipulaties uit en bestaat uit de rekeneenheid zelf (ALU), de accumulator, de carryvlag en de instructie-decoder.
Gewoonlijk wordt in de accumulator aanwezige data in de ALU gecombineerd met data op de interne bus (die uit een andere bron, bijvoorbeeld een register of I/O-poort afkomstig is), waarna het resultaat in de accumulator of een ander register wordt opgeborgen.

- ALU
  De ALU accepteert 8-bit datawoorden uit een of twee bronnen en genereert onder besturing van de instructie-decoder een 8-bit resultaat. De ALU kan de volgende functies uitvoeren:
  - Optellen (ADD) met of zonder carry
  - AND, OR, EXOR
  - Met 1 verhogen of verlagen (increment/decrement)
  - Bit complement

- Naar links of naar rechts roteren
- Halve bytes verwisselen (nibble swap)
- BCD omzetten in decimaal (BCD decimal adjust)

Indien het resultaat van een ALU-operatie groter dan 8 bit wordt (overflow van het belangrijkste bit), wordt in het Programstatuswoord de "carry-flag" geset.

- Accumulator
  De accumulator (accu) is het belangrijkste register van de processor.
  Hij is een van de toegangen tot de ALU en wordt vaak gebruikt om het resultaat van een bewerking door de ALU op te slaan. Data van en naar de I/O-poorten en geheugen gaat meestal ook via de accu.

- Instructie-decoder
  Het operatie-code (op code) deel van elke instructie wordt opgeslagen in de instructie-decoder en omgezet in signalen die de functies van de verschillende blokken van de ALU besturen. Deze lijnen regelen de herkomst van de data, het bestemmingsregister en de door de ALU uitgevoerde functie.

Programmageheugen
Het programmageheugen kan 1024, 2048 of 4096 8-bit brede woorden bevatten en wordt geadresseerd door de programmateller (program counter).
In de 8748 en 8749 is dit geheugen een EPROM die door de gebruiker zelf kan worden geprogrammeerd en gewist. In de 8048, 8049 en 8050 is het geheugen een in de fabriek geprogrammeerde (mask programmable) ROM.
De 8035, 8039 en 8040 hebben geen intern programmageheugen en moeten daarom externe geheugens gebruiken. De programmacode is voor alle versies identiek. Om de bovenste 2 kB van het programma-geheugen in de 8050 te bereiken moeten een "select memory bank" en een JUMP of CALL instructie worden uitgevoerd om de 2 kB drempel te overschrijden.

Zoals ook in figuur 7/6.1-2 wordt getoond heeft het programmageheugen drie speciale gedeelten.
- Lokatie 0
  Wanneer de resetlijn van de processor wordt geaktiveerd, wordt de eerste instructie uit lokatie 0 gehaald.
- Lokatie 3
  Het aktiveren van de interruptlijn van de processor (bij vrijgegeven interruptie) veroorzaakt een jump naar de subroutine die op lokatie 3 begint.
- Lokatie 7
  Wanneer een timer/counter bij overflow een interrupt veroorzaakt (indien vrijgegeven), wordt naar de subroutine gesprongen die op lokatie 7 begint.

Daarom is de eerste instructie die na initialisatie moet worden uitgevoerd in lokatie 0 opgeslagen, bevat lokatie 3 het eerste woord van een externe interrupt-service subroutine en bevindt het eerste woord van een timer/counter-service subroutine zich in lokatie 7.

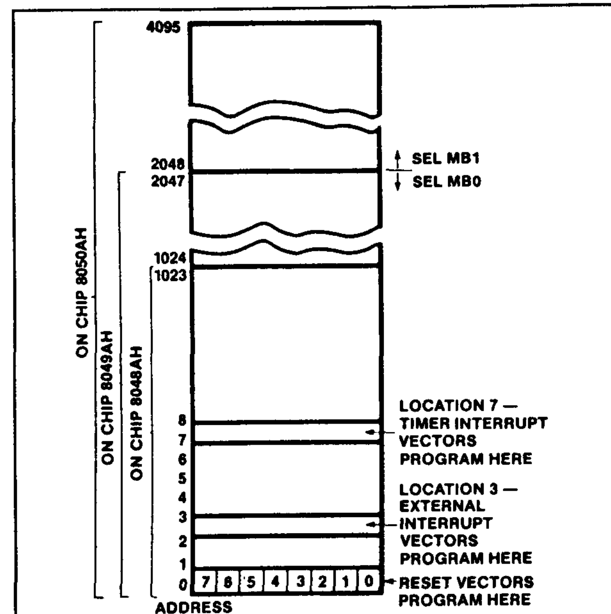
Het programma-geheugen kan zowel constanten als programma-instructies bevatten. Met instructies als MOVP en MOVP3 kan bijvoorbeeld gemakkelijk toegang worden verkregen tot data "lookup" tabellen.

Datageheugen
Het interne datageheugen is georganiseerd als 64, 128 of 256 8-bit brede woorden. Alle lokaties kunnen indirect met behulp van een van beide RAM pointer-registers worden geadresseerd die op de adressen 0 en 1 van het register-array zijn gehuisvest.
Bovendien zijn, zoals in figuur 7/6.1-3 te zien is, de eerste 8 plaatsen in het array (0 tot en met 7) bestemd als werkregisters en kunnen door verschillende instructies direct worden geadresseerd.

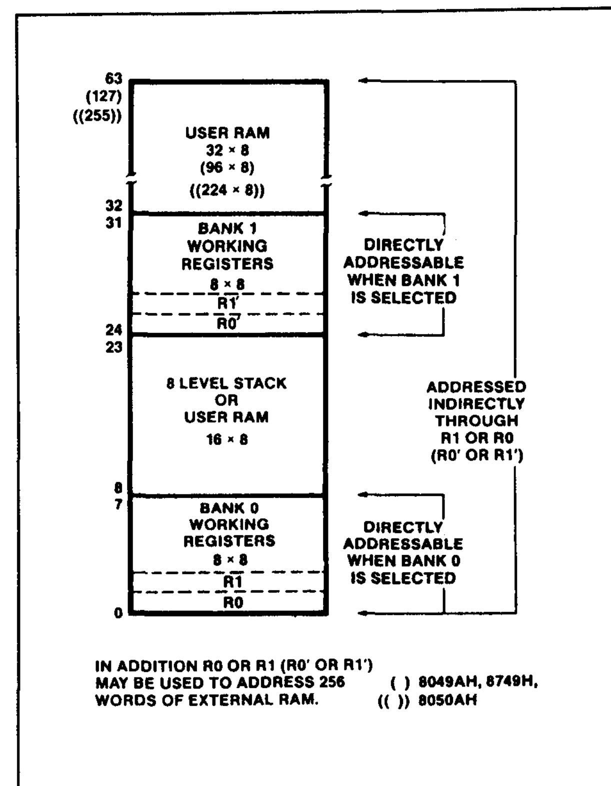
6.1 8-bit microcontrollers van de 8048-familie

Aangezien deze registers gemakkelijker bereikbaar zijn, worden zij meestal toegepast om vaak gebruikte tussenresultaten op te slaan.
De DNJZ instructie kan de werkregisters zeer efficiënt als looptellers gebruiken door de programmeur in staat te stellen het register met een instructie te decrementeren en te testen.

Door het uitvoeren van een Register Bank Switch instructie (SEL RB) worden niet de RAM-lokaties 0 tot en met 7 aangewezen als werkregisters, maar de lokaties 24 tot en met 31 en deze zijn dan direct adresseerbaar.
Deze tweede groep werkregisters kan als uitbreiding van de eerste groep dienen of worden gereserveerd voor interrupt-service subroutines, zodat de registers van groep 0 die door het hoofdprogramma worden gebruikt ogenblikkelijk door een Bank Switch worden gered.
Wanneer deze tweede groep niet wordt gebruikt, kunnen de plaatsen 24 tot en met 31 toch als gewone RAM worden geadresseerd.
Aangezien de twee RAM pointer-registers R0 en R1 deel uitmaken van de werkregisterbank, creëert een bankverwisseling nog twee pointer-registers (R0' en R1'), waardoor op eenvoudige wijze tegelijkertijd vier aparte werkgebieden in de RAM kunnen worden bereikt.
De RAM-plaatsen 8 tot en met 23 hebben eveneens een dubbele rol, aangezien zij de programma stackpointer bevatten, zoals verderop wordt uitgelegd.
Deze lokaties worden zowel tijdens subroutine calls door de Stack Pointer geadresseerd als door de RAM pointerregisters R0 en R1.
Als het aantal subroutine-nestingen minder dan 8 is, zijn niet alle stackregisters nodig en kunnen deze als gewone RAM-plaatsen worden gebruikt.
Elke niet-gebruikte subroutine-nesting levert de gebruiker twee extra RAM-plaatsen op.



Figuur 7/6.1-2: Indeling van het programmageheugen.

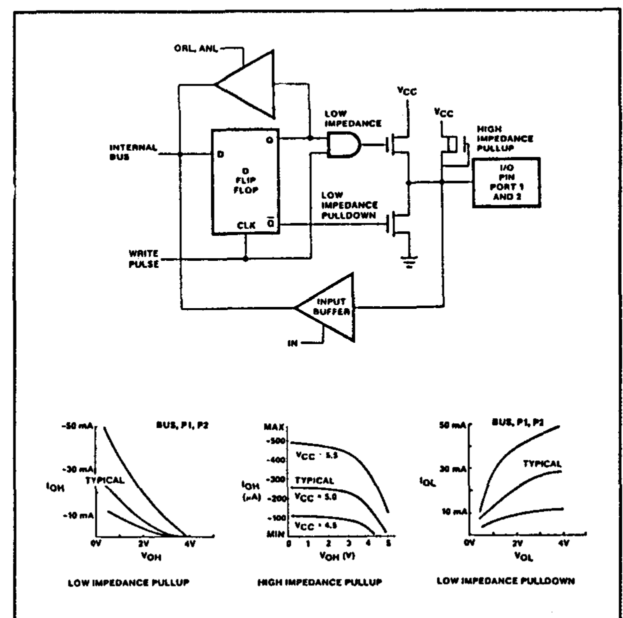


Figuur 7/6.1-3: Indeling van het datageheugen.

In- en Uitgangen

De 8048 heeft 27 lijnen die voor ingangs- of uitgangs-functies kunnen worden gebruikt. Deze lijnen zijn verdeeld over drie poorten met elk 8 lijnen die zowel ingang, uitgang als bidirectioneel kunnen zijn en drie "test"-ingangen waarmee het verloop van het programma kan worden veranderd als ze door voorwaardelijke sprong (conditionele jump)-instructies worden afgevraagd.

- Poort 1 en poort 2
  De poorten 1 en 2 zijn beide 8 bit breed en hebben identieke eigenschappen. Data die naar deze poorten wordt geschreven, wordt statisch gelatcht en blijft onveranderd tot hij wordt overschreven. Wanneer deze lijnen als ingangen worden gebruikt, hebben zij geen latch, hetgeen wil zeggen dat de ingangssignalen aanwezig moeten blijven tot ze door een ingangsinstructie worden gelezen. De poorten zijn als ingang volledig TTL-compatibel en als uitgang kunnen zij een standaard TTL-belasting aandrijven (fan-out = 1).
  De lijnen van de poorten 1 en 2 worden quasi- bidirectioneel genoemd, doordat zij beschikken over een speciale structuur die elke lijn in staat stelt zowel als ingang, uitgang of beide te dienen, terwijl de uitgangen statisch gelatcht worden.
  Figuur 7/6.1-4 toont alle details van deze schakeling. Elke lijn ligt voortdurend via een vrij hoog-impedante weerstand aan V_{CC}. Dit is voldoende om bij TTL-HOOG niveau de source-stroom te leveren, terwijl hij toch LAAG kan worden getrokken door een standaard TTL-poort. Op deze manier kan dezelfde pen als ingang en als uitgang worden gebruikt. Om bij LAAG-naar-HOOG overgangen korte schakeltijden te verkrijgen, wordt telkens wanneer een "1" naar een lijn wordt geschreven tijdelijk (1/5 van de machine-cyclus) een relatief laag-impedant onderdeel ingeschakeld. Wanneer een "0" naar de lijn wordt geschreven, maakt een laag-impedant onderdeel de lichte optrekking ongedaan en levert voldoende sinkstroom voor TTL. Aangezien de neertrek (pulldown) transistor een laag-impedante schakeling is, moet op elke lijn die als ingang moet werken eerst een "1" worden geschreven. Met een reset worden alle lijnen in de hoog-impedante "1"-toestand geïnitialiseerd.



Figuur 7/6.1-4: "Quasi-bidirectionele" poortstructuur.

Let op dat ORL en ANL lees/schrijf-operaties zijn. Worden zij uitgevoerd dan "leest" de CPU de data op de poort, verandert deze overeenkomstig de instructie en "schrijft" de data daarna weer naar de poort. Het "schrijven" (dat feitelijk een OUTL-instructie is) aktiveert heel even de laag-impedante optrek, zelfs als de data "1" was en niet veranderde. Hiermee moet vooral rekening worden gehouden als in- en uitgangen op dezelfde poort zijn aangesloten.

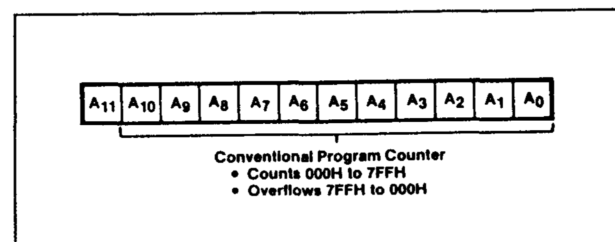
- Bus
  BUS is ook een echte bidirectionele 8-bit poort met bijbehorende in- en uitgangsstrobes. Als de bidirectionele eigenschappen niet worden gebruikt kan BUS dienen

als statisch gelatchte uitgangspoort of als niet-latchende ingangspoort. Bij deze poort kunnen in- en uitgangslijnen echter niet worden gemengd.
Bij toepassing als statische poort wordt data geschreven en gelatcht door de OUTL-instructie en ingelezen door de INS-instructie. Beide instructies genereren pulsen op de overeenkomstige $\overline{RD}$ en $\overline{WR}$ strobe-lijnen.
In de statische poort-mode worden zij echter meestal niet gebruikt. Bij toepassing als bidirectionele poort worden de MOVX-instructies gebruikt om data te lezen of te schrijven.
Bij het schrijven wordt een negatieve puls op de $\overline{WR}$-uitgang gegenereerd en is de data geldig op de (stijgende) achterflank van $\overline{WR}$. Bij uitlezen van de poort wordt een negatieve puls op de $\overline{RD}$-uitgang gegenereerd en moet data geldig blijven tot op de achterflank van $\overline{RD}$. Wanneer noch gelezen noch geschreven wordt bevinden de BUS-lijnen zich in een hoog-impedante toestand.
Bij controllers met extern geheugen (8035, 8039 en 8040) word BUS gebruikt om data en adressen uit te wisselen. Instructies die op BUS betrekking hebben mogen dan niet worden gebruikt.

- Test- en INT-ingangen
  De drie ingangspennen T0, T1 en $\overline{INT}$ kunnen door middel van een conditionele jump getest worden. Deze pennen maken het mogelijk dat het programma door signalen wordt bestuurd zonder dat een ingangspoort in de accu geladen hoeft te worden. De T0, T1 en $\overline{INT}$-pennen hebben ook nog andere functies (zie bij beschrijving van de aansluitingen).

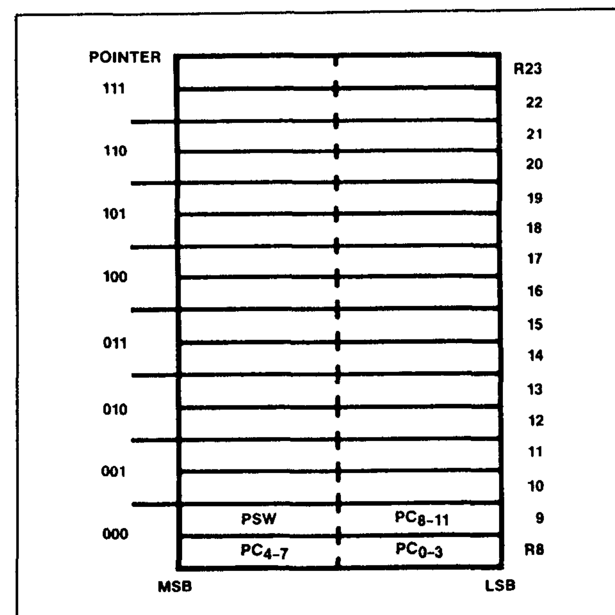
Programmateller en Stack

De programmateller (Program Counter) is een onafhankelijke teller, terwijl het programmateller geheugen (Program Counter Stack) gebruik maakt van registerparen in het datageheugen. Van de programmateller worden slechts 10, 11 of 12 bit gebruikt om de 1024, 2048 of 4096 woorden van het op de chip aanwezige programmageheugen te adresseren. De hoogste bits kunnen voor een extern programmageheugen worden gebruikt (zie figuur 7/6.1-5). De programmateller wordt door aktiveren van de resetlijn op nul gezet.



Figuur 7/6.1-5: Programmateller.

Door een interrupt of een CALL naar een subroutine wordt de inhoud van de programmateller in een van de 8 registerparen van de stack opgeslagen, zoals in figuur 7/6.1-6 te zien is. Welk paar gebruikt wordt is afhankelijk van een 3-bit stackpointer die deel uitmaakt van het programma-status woord (PSW).



Figuur 7/6.1-6: Programmateller-geheugen: Stack.

6.1 8-bit microcontrollers van de 8048-familie

De RAM-lokaties 8 tot en met 23 zijn beschikbaar als stackregister en worden gebruikt voor opslag van de programmateller en 4 bit van de PSW. Als de stackpointer in de beginstand 000 staat, wijst hij naar de lokaties 8 en 9. De eerstvolgende interrupt of sprong naar een subroutine maakt dat de inhoud van de programmateller naar de lokaties 8 en 9 van de RAM wordt overgebracht. De stackpointer wordt dan met 1 verhoogd om (in afwachting van een volgende CALL) de lokaties 10 en 11 aan te wijzen. Hierdoor kan het "nesten" van subroutines in subroutines doorgaan tot 8 maal, zonder dat de stack overloopt. Als een overloop (overflow) optreedt, wordt het diepst weggeborgen adres (op de lokaties 8 en 9) overschreven omdat de stackpointer van 111 overloopt naar 000 (hij loopt ook "onder" van 000 tot 111).

Het einde van een subroutine (dat wordt aangegeven door een return-instructie RET of RETR) heeft tot gevolg dat de stackpointer met 1 wordt verlaagd en dat de inhoud van het aangewezen registerpaar naar de programmateller wordt overgebracht.

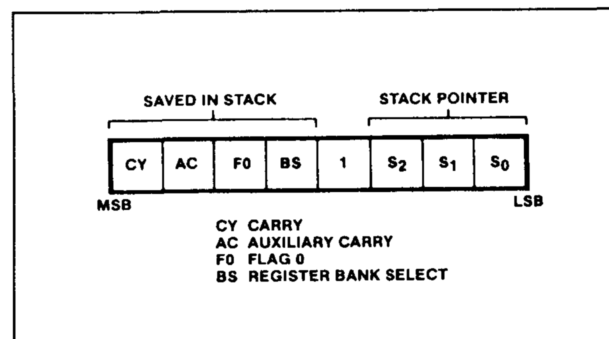
Programma-statuswoord

Het statuswoord (program status word PSW) is een 8-bit register dat via de accumulator kan worden geladen en uitgelezen.
Figuur 7/6.1-7 laat zien welke informatie verkrijgbaar is. Het programma-statuswoord is eigenlijk een over de processor verdeelde reeks flip-flop's die in zijn geheel kan worden uitgelezen of beschreven. Doordat de PSW beschreven kan worden, kan de status van de controller na een power-down toestand gemakkelijk worden hersteld.

De bovenste 4 bit van de PSW worden bij elke aanroep van een subroutine of een interrupt vector in de stack opgeslagen en eventueel bij terugkeer met de RETR-instructie weer teruggebracht. De RET-instructie brengt de PSW niet op orde.

De PSW-bits hebben de volgende betekenis:

- Bits 0 tot en met 2
  Stackpointer-bits (S0, S1 en S2)
- Bit 3
  Niet gebruikt ("1" bij uitlezen)
- Bit 4
  Schakelbit van de werkregistergroep (Working Register Bank Switch - BS)
  0 = bank 0
  1 = bank 1
- Bit 5
  Door de gebruiker instelbare Flag 0 bit (F0); kan gecomplementeerd of gecleared worden en wordt getest met de voorwaardelijke sprongopdracht JFO
- Bit 6
  Carry-bit (Auxiliary Carry - AC) dat ontstaat door een ADD-instructie en wordt gebruikt bij de decimal adjust instructie DAA
- Bit 7
  Carry-vlag (CY) die aangeeft dat de vorige operatie een overflow van de accu heeft veroorzaakt



Figuur 7/6.1-7: Programma-statuswoord (PSW).

Logika voor voorwaardelijke sprongen

De logika voor voorwaardelijke sprongopdrachten (conditional branch logic) in de processor maakt het mogelijk om verschillende interne en externe condities met het gebruikersprogramma te testen. Door de voorwaardelijke sprongopdracht te gebruiken kunnen de condities die in tabel 7/6.1-2 zijn vermeld het verloop van het programma beïnvloeden.

6.1 8-bit microcontrollers van de 8048-familie

Device Testable	Jump Conditions (Jump On)	
	All zeros	not all zeros
Accumulator	All zeros	not all zeros
Accumulator Bit	—	1
Carry Flag	0	1
User Flags (F0, F1)	—	1
Timer Overflow Flag	—	1
Test Inputs (T0, T1)	0	1
Interrupt Input ($\overline{INT}$)	0	—

Tabel 7/6.1-2: Condities die een voorwaardelijke sprong kunnen veroorzaken.

Interrupties

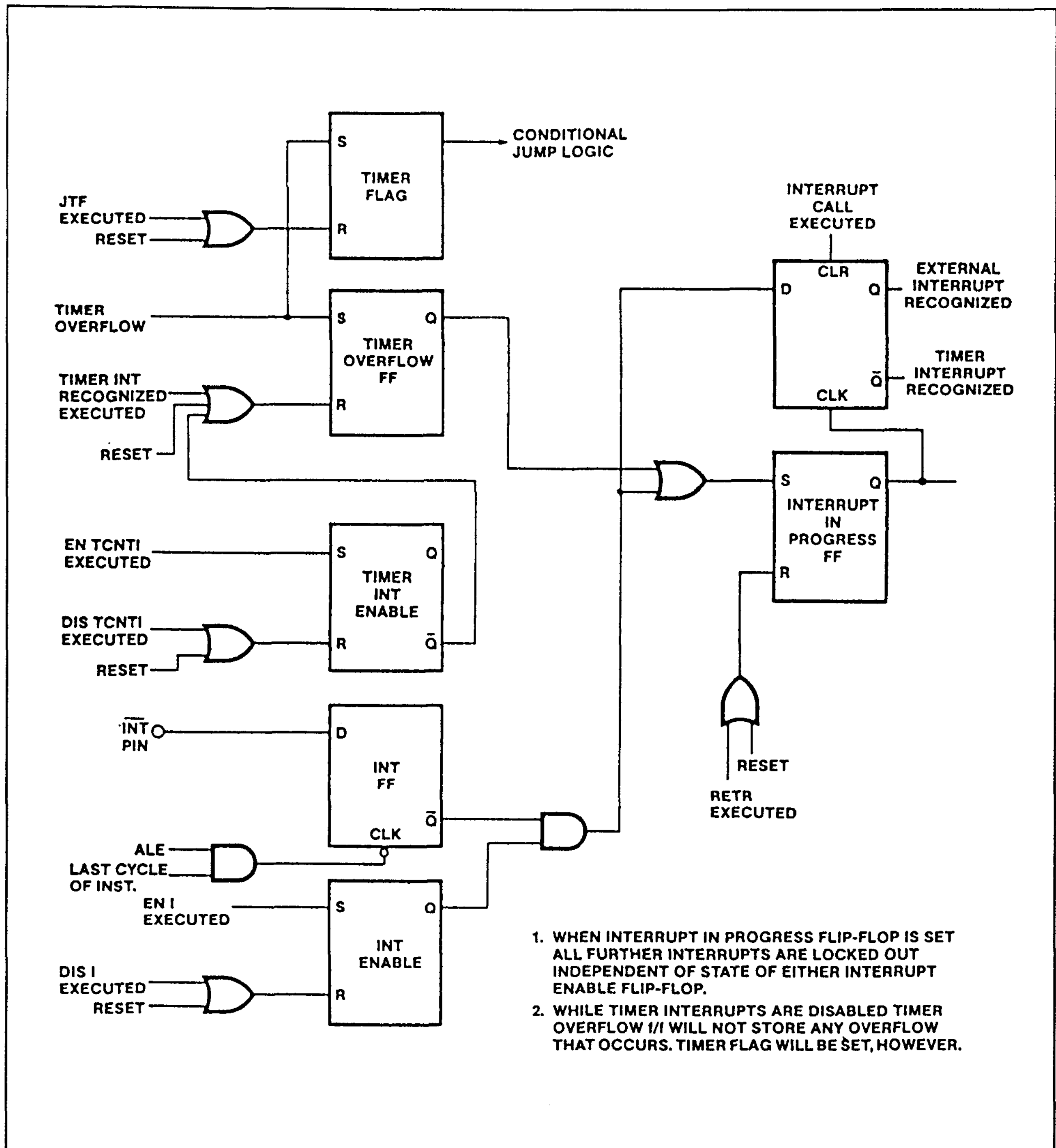
Door een LAAG niveau op de $\overline{INT}$-pen te zetten wordt een interruptie-volgorde gestart. De interrupt-ingang is niveau-getriggerd en aktief-LAAG om "wired-OR" van verschillende bronnen mogelijk te maken. In figuur 7/6.1-8 wordt een overzicht gegeven van de interrupt logika van de 8048. De interruptlijn wordt bij elke instructiecyclus bemonsterd.
Wanneer een LAAG wordt gedetecteerd veroorzaakt dit een "call naar subroutine" op lokatie 3 van het programmageheugen zodra alle cyclussen van de lopende instructie zijn uitgevoerd. Bij 2-cyclus instructies wordt de interruptlijn alleen op de tweede cyclus bemonsterd. $\overline{INT}$ moet tenminste gedurende drie machinecyclussen LAAG worden gehouden om correcte interrupties te garanderen. Zoals bij elke CALL naar subroutine worden de programmateller en de programma-status in de stack bewaard.

Geheugenplaats 3 bevat meestal een onvoorwaardelijke sprong naar een interrupt-service subroutine die ergens anders in het programmageheugen staat. Het einde van een interrupt-service subroutine wordt gesignaleerd door de uitvoering van een Return en Restore instructie RETR. Het interruptiesysteem heeft slechts één niveau, hetgeen wil zeggen dat alle verdere interrupt-verzoeken tot na de uitvoering van een RETR worden genegeerd wanneer er reeds een is gedetecteerd. Dit gebeurt bij het begin van de tweede cyclus van de RETR-instructie. Deze volgorde geldt ook voor een interne interrupt die door een timer-overflow wordt veroorzaakt. Wanneer een interrupt van een interne timer/counter en een externe interrupt tegelijk worden gedetecteerd, gaat de externe voor (zie ook bij het timer/counter gedeelte). Indien nodig kan een tweede externe interrupt worden gecreëerd door de timer/counter interrupt vrij te geven.
De teller moet dan met FFH worden geladen (FF hexadecimaal = 1 minder dan overflow) en in de event counter mode worden gezet. Een "1" naar "0" overgang op de T1-ingang zal dan een interruptvector naar lokatie 7 veroorzaken.

Interrupt timing

De interrupt-ingang kan door het programma worden vrijgegeven of gesperd door de EN1 en DIS1 instructies te gebruiken. Interrrupties worden door Reset gesperd en blijven zo totdat zij door het gebruikersprogramma worden vrijgegeven. Een interrupt request moet worden weggehaald voordat de RETR-instructie wordt uitgevoerd, omdat de processor anders direct weer in de service-routine zal komen. Bij veel randapparaten wordt deze situatie voorkomen doordat de interrupt-requestlijn wordt gereset zodra de processor toegang krijgt tot het databufferregister (lezen of schrijven). Wanneer het interrumperende apparaat niet door de processor behandeld hoeft te worden, kan een uitgangslijn van de 8048 worden aangewezen om de interruptie te bevestigen (interrupt acknowledge). Deze lijn wordt dan geaktiveerd door de service subroutine waarmee het interrupt-request wordt gereset. De $\overline{INT}$-pen kan ook worden getest door de voorwaardelijke sprongopdracht JNI te gebruiken.
Deze instructie kan worden gebruikt om de aanwezigheid van een "hangende" interrupt te detecteren voordat de interrupts zijn vrijgegeven. Als een interrupt wordt gesperd, kan $\overline{INT}$ als test-ingang worden gebruikt (net als T0 en T1).

6.1 8-bit microcontrollers van de 8048-familie

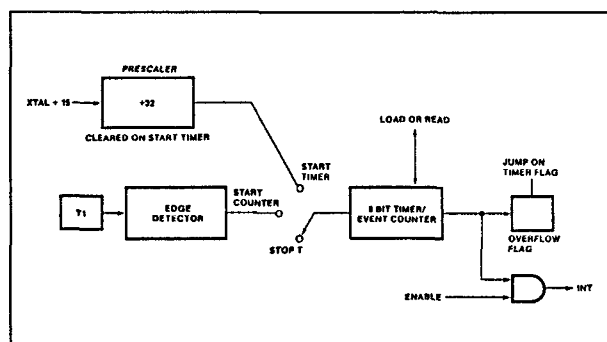


Figuur 7/6.1-8: Interrupt logika.

Timer/Counter

De 8048 heeft een teller die de gebruiker behulpzaam is bij het tellen van externe verschijnselen en die precieze tijdvertragingen kan opwekken zonder daarmee de processor lastig te vallen. In beide bedrijfsmodes is de werking van de teller dezelfde en is de bron van het ingangssignaal het enige verschil. De opzet van de timer/gebeurtenissenteller wordt getoond in figuur 7/6.1-9.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-9: Blokschema van de timer/gebeurtenissenteller.

Teller

De 8-bit binaire teller kan met twee MOV-instructies worden gepreset en uitgelezen. Deze instructies brengen de inhoud van de accumulator over naar de teller, en omgekeerd. De inhoud van de teller kan door Reset worden beïnvloed en dient dus door de software geïnitialiseerd te worden. Het tellen wordt gestopt door een Reset of een STOP TCNT-instructie. Het tellen begint weer als timer met een START T-instructie of als gebeurtenissenteller met een START CNT-instructie. Als de teller eenmaal gestart is, telt hij op tot het maximum FF, loopt over naar 00 en gaat net zo lang door totdat hij gestopt wordt door een STOP CNT-instructie of Reset.

Door de verhoging van maximum naar nul (overflow) wordt een overflow-flag flip-flop geset en een interrupt request gegenereerd. De toestand van de overflow-flag kan met de voorwaardelijke sprongopdracht worden getest.
De flag wordt gereset door een JTF uit te voeren of door Reset.
Het interrupt request wordt in een latch opgeslagen en vervolgens gecombineerd (OR-functie) met de externe interrupt-ingang $\overline{\text{INT}}$.
De timer interrupt kan met de DIS TCNT1 en EN TCNT1 instructies onafhankelijk van de externe interrupt worden gesperd of vrijgegeven. Als hij is vrijgegeven zal het overlopen van de teller een subroutine call op lokatie 7 veroorzaken, waar het begin van de serviceroutine van de teller of de timer kan worden opgeslagen.

Als interrupts door de timer en door externe bronnen tegelijk optreden, zal de externe bron worden herkend en wordt naar lokatie 3 gesprongen. Aangezien de interrupt van de timer in een latch wordt opgeslagen blijft deze "hangen" totdat het externe apparaat is bediend. Direct bij terugkeer uit de serviceroutine wordt de interrupt van de timer gezien en wordt deze gereset door een sprong naar lokatie 7. De timer interrupt kan natuurlijk ook worden opgeheven met een DIS TCNT1-instructie.

- Gebruik als gebeurtenissen-teller
  Door het uitvoeren van een START CNT-instructie wordt de T1-ingang met de ingang van de teller verbonden en wordt de teller vrijgegeven. De T1-ingang wordt bemonsterd aan het begin van S3 in de instructiecyclus (zie figuur 7/6.1-11 en tabel 7/6.1-4).
  Bij nieuwere versies gebeurt dit op S4. De teller zal door opeenvolgende HOOG-naar-LAAG overgangen op T1 worden verhoogd. T1 moet hierbij gedurende tenminste 1 machinecyclus LAAG worden gehouden om er zeker van te zijn dat hij niet gemist wordt.
  De maximale snelheid waarmee de teller kan worden verhoogd is eenmaal per drie instructie-cyclussen (bij gebruik van een 8 MHz kristal elke 5,7 µs).
  Er is echter geen minimale telfrequentie. De T1-ingang moet na elke overgang gedurende tenminste 1/5 machinecyclus HOOG blijven.

- Gebruik als timer
  Door het uitvoeren van een START T-instructie wordt een interne clock op de ingang van de teller aangesloten en wordt de teller vrijgegeven.
  De interne clock wordt via een :32-prescaler afgeleid van de machine-clock.

6.1 8-bit microcontrollers van de 8048-familie

Tijdens de START T-instructie wordt de prescaler gereset.
De hierdoor gevormde clock verhoogt de teller eenmaal per 32 machinecyclussen. Door de teller te presetten en de overflow te detecteren kunnen verschillende vertragingen van 1 tot 256 telpulsen worden verkregen.
Langere tijden dan 256 telpulsen zijn mogelijk door meerdere overflows onder software besturing in een register op te tellen. Door een externe clock op de T1-ingang aan te sluiten en de teller in de gebeurtenissenteller mode te zetten kunnen tijdseenheden van minder dan 1 telpuls worden verkregen. ALE gedeeld door 3 of meer kan als deze externe clock dienen. Zeer korte vertragingen of "fijn afstemmen" van grotere vertragingen kunnen gemakkelijk met behulp van software vertragingslussen worden verkregen.
Dikwijls is een seriële verbinding met een lid uit de 8048-familie gewenst. In tabel 7/6.1-3 wordt een overzicht gegeven van de aantallen telpulsen en cyclussen die bij bepaalde kristalfrequenties nodig zijn voor het verkrijgen van enkele specifieke baud rates.

Clock en timing schakelingen

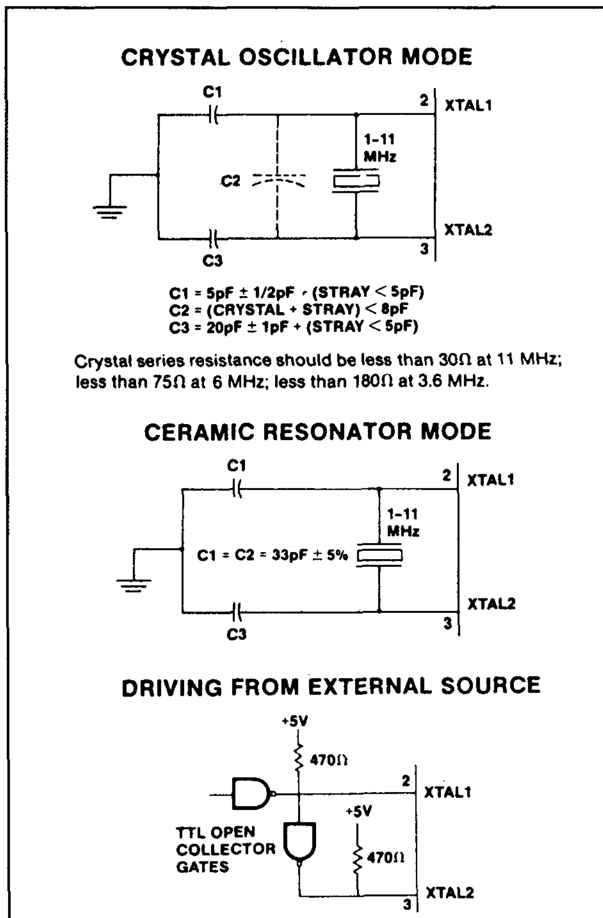
Met uitzondering van een referentie-frequentie (een kristal, keramische resonator of externe frequentiebron) heeft de 8048 voor het opwekken van timing-signalen geen externe componenten nodig.

- Oscillator
  De op de chip aanwezige oscillator is een parallelle resonantieschakeling met hoge versterking voor het frequentiegebied tussen 1 en 11 MHz. De XTAL1-pen is de ingang naar de versterkertrap, terwijl XTAL2 de uitgang is. Om de schakeling te laten oscilleren moet een kristal of een keramische resonator tussen XTAL1 en XTAL2 worden aangesloten (zie figuur 7/6.1-10). Voor precieze clock-tijden moet een kristal worden gebruikt, terwijl een keramische resonator een minder nauwkeurige frequentie oplevert. Ook kan een extern opgewekte clock op XTAL1 en XTAL2 worden aangesloten.

Frequency (MHz)		**T_{cy}**	**T0 Prr(1/5 T_{cy})**	**Timer Prescaler (32 T_{cy})**
4		3.75μs	750ns	120μs
6		2.50μs	500ns	80μs
8		1.88μs	375ns	60.2μs
11		1.36μs	275ns	43.5μs
Baud Rate	4 MHz Timer Counts + Instr. Cycles	6 MHz Timer Counts + Instr. Cycles	8 MHz Timer Counts + Instr. Cycles	11 MHz Timer Counts + Instr. Cycles
110	75 + 24 Cycles .01% Error	113 + 20 Cycles .01% Error	151 + 3 Cycles .01% Error	208 + 28 Cycles .01% Error
300	27 + 24 Cycles .1% Error	41 + 21 Cycles .03% Error	55 + 13 Cycles .01% Error	76 + 18 Cycles .04% Error
1200	6 + 30 Cycles .1% Error	10 + 13 Cycles .1% Error	12 + 27 Cycles .06% Error	19 + 4 Cycles .12% Error
1800	4 + 20 Cycles .1% Error	6 + 30 Cycles .1% Error	9 + 7 Cycles .17% Error	12 + 24 Cycles .12% Error
2400	3 + 15 Cycles .1% Error	5 + 6 Cycles .4% Error	6 + 24 Cycles .29% Error	9 + 18 Cycles .12% Error
4800	1 + 23 Cycles 1.0% Error	2 + 19 Cycles .4% Error	3 + 14 Cycles .74% Error	4 + 25 Cycles .12% Error

Tabel 7/6.1-3: Instellingen van de teller voor enkele baud rates.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-10: Voor de oscillator is alleen een kristal, een keramische resonator of een externe frequentiebron nodig.

- Toestandsteller
  Het uitgangssignaal van de oscillator wordt in de toestandsteller door 3 gedeeld om een clock te verkrijgen die de toestandstijden van de machine bepaalt (CLK). CLK kan op de aansluitpen T0 worden gezet door een ENT0 CLK-instructie uit te voeren.
  De uitgang van CLK op T0 wordt gesperd door Reset van de processor (zie figuur 7/6.1-11).

- Cyclusteller
  Hierna wordt CLK in de cyclusteller door 5 gedeeld om een clock te verkrijgen die een uit 5 machine-toestanden bestaande machine-cyclus mogelijk maakt (zie ook figuur 7/6.1-11). Tabel 7/6.1-4 laat zien hoe de verschillende interne operaties over de machine-toestanden worden verdeeld.
  Deze clock wordt door zijn functie in 8048-systemen met extern geheugen ook wel Address Latch Enable (ALE) genoemd. ALE is continu aanwezig op de ALE uitgangspen.

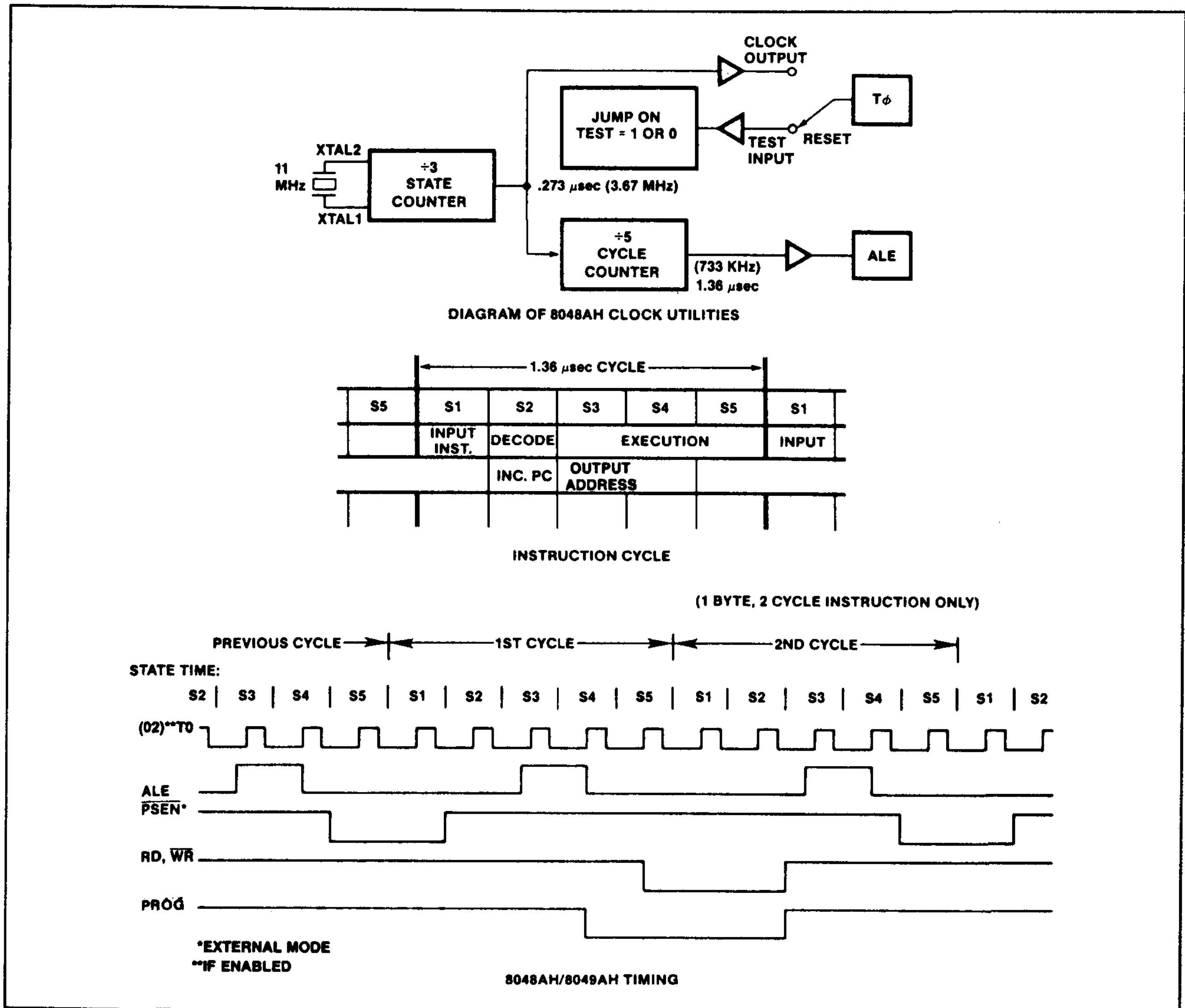
Reset

De $\overline{\text{RESET}}$-ingang maakt initialisatie van de processor mogelijk.
De Schmitt-trigger ingang heeft een interne optrekvoorziening die in combinatie met een externe condensator van 1 µF een inwendige resetpuls opwekt die lang genoeg is om te garanderen dat alle schakelingen worden gereset (figuur 7/6.1-12).

Wanneer de resetpuls extern wordt gegenereerd, moet de $\overline{\text{RESET}}$-pen gedurende tenminste 10 ms na inschakeling van de voeding LAAG worden gehouden. Wanneer de voedingsspanning reeds aanwezig is en de oscillator stabiel werkt, zijn slechts 5 machine-cyclussen nodig (bijvoorbeeld 6,8 µs bij 11 MHz).
Tijdens het resetten zijn ALE en $\overline{\text{PSEN}}$ (als EA = 1) aktief.

Het resetten vervult de volgende functies:
- De programmateller wordt op nul gezet.
- De stackpointer wordt op nul gezet.
- Registerbank 0 wordt gekozen.
- Geheugenbank 0 wordt gekozen.
- BUS wordt in de hoog-impedante toestand gezet (behalve wanneer EA = 5 V).
- Poort 1 en poort 2 worden als ingang geschakeld.
- (Timer en externe) interrupties worden gesperd.
- De timer wordt gestopt.
- De timer-flag wordt leeggemaakt.
- F0 en F1 worden leeggemaakt.
- CLK wordt losgemaakt van T0.

6.1 8-bit microcontrollers van de 8048-familie



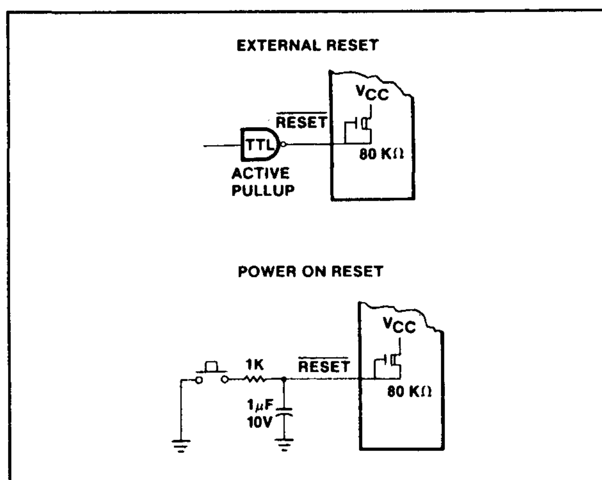
Figuur 7/6.1-11: Opwekking van de inwendige timing en cyclus-timing van de 8048.

6.1 8-bit microcontrollers van de 8048-familie

	CYCLE 1					CYCLE 2				
INSTRUCTION	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
IN A,P	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	—	—	READ PORT	—	• —	—
OUTL P,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	OUTPUT TO PORT	—	—	—	• —	—
ANL P, * DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
ORL P, * DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
INS A, BUS	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	—	—	READ PORT	—	• —	—
OUTL BUS, A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	INCREMENT TIMER	OUTPUT TO PORT	—	—	—	• —	—
ANL BUS, * DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
ORL BUS, * DATA	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	*INCREMENT TIMER	READ PORT	FETCH IMMEDIATE DATA	—	INCREMENT PROGRAM COUNTER	*OUTPUT TO PORT	—
MOVX @ R,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT RAM ADDRESS	INCREMENT TIMER	OUTPUT DATA TO RAM	—	—	—	• —	—
MOVX A,@R	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT RAM ADDRESS	INCREMENT TIMER	—	—	READ DATA	—	• —	—
MOVD A,P_i	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	—	—	READ P2 LOWER	—	• —	—
MOVD P_i,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA TO P2 LOWER	—	—	—	• —	—
ANLD P,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA	—	—	—	• —	—
ORLD P,A	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	OUTPUT OPCODE/ADDRESS	INCREMENT TIMER	OUTPUT DATA	—	—	—	• —	—
J(CONDITIONAL)	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	SAMPLE CONDITION	*INCREMENT SAMPLE	—	FETCH IMMEDIATE DATA	—	UPDATE PROGRAM COUNTER	• —	—
STRT T STRT CNT	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	• —	START COUNTER					
STOP TCNT	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	• —	STOP COUNTER					
ENI	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	• ENABLE INTERRUPT	—					
DIS I	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER	—	• DISABLE INTERRUPT	—					
ENTO CLK	FETCH INSTRUCTION	INCREMENT PROGRAM COUNTER		• ENABLE CLOCK	—					

*VALID INSTRUCTION ADDRESSES ARE OUTPUT AT THIS TIME IF EXTERNAL PROGRAM MEMORY IS BEING ACCESSED.
(1) IN LATER MCS-48 DEVICES T1 IS SAMPLED IN S4.

Tabel 7/6.1-4: Timing van verschillende instructies ten opzichte van de machine-toestanden S1 tot en met S5.



Figuur 7/6.1-12: De externe resetschakelingen.

Single-step

Zoals figuur 7/6.1-13 laat zien heeft de gebruiker door deze voorziening de mogelijkheid stap-voor-stap door het programma te lopen om eventuele fouten op te sporen. Terwijl de processor gestopt is, staat het adres van de volgende instructie reeds in BUS en de lage helft van poort 2 te wachten. De gebruiker kan het programma zodoende stapsgewijs volgen. In figuur 7/6.1-13 is tevens een tijddiagram opgenomen, waarin de interaktie tussen de ALE-uitgang en de $\overline{SS}$-ingang te zien is. De inhoud van de BUS-buffer gaat bij de stap-voor-stap bewerking verloren. Indien nodig kan een extra latch

echter uitkomst bieden. Data is bij de voorflank van ALE geldig.

De 8048 werkt als volgt in de stap-voor-stap mode:

- De processor wordt door een LAAG niveau op $\overline{SS}$ gevraagd te stoppen.
- De processor reageert door tijdens het adres-fetch gedeelte van de volgende instructie te stoppen. Indien bij ontvangst van het stap-voor-stap commando een dubbele cyclus-instructie bezig is, worden beide cyclussen afgemaakt voordat gestopt wordt.
- De processor bevestigt dat hij gestopt is door ALE HOOG te maken. In deze toestand (die voor onbepaalde tijd kan worden gehandhaafd) is het adres van de volgende instructie die moet worden opgehaald aanwezig op BUS en de lage helft van poort 2.
- $\overline{SS}$ wordt dan HOOG gemaakt om de processor uit de gestopte toestand te brengen en hem in staat te stellen de volgende instructie op te halen. Het verlaten van stop wordt aangegeven door het LAAG gaan van ALE.
- Om de processor bij de volgende instructie te laten stoppen moet $\overline{SS}$ weer LAAG worden gemaakt, kort nadat ALE LAAG gaat. Als $\overline{SS}$ HOOG blijft, gaat de processor verder in de "run" mode.

In figuur 7/6.1-13 is een schema getekend, waarmee een 8048 microcontroller in de stap-voor-stap mode kan worden gebruikt. De D-type flip-flop met preset en clear wordt gebruikt om $\overline{SS}$ op te wekken. In de run-mode wordt $\overline{SS}$ HOOG gehouden door de flip-flop in de preset-toestand te houden (preset gaat voor clear).
Om in de stap-voor-stap mode te komen, wordt preset verwijderd, zodat ALE $\overline{SS}$ LAAG kan maken via de clear-ingang. ALE moet worden gebufferd aangezien de clear-ingang van een 7474 flip-flop equivalent is met 3 TTL-belastingen. De processor bevindt zich nu in de gestopte toestand. De volgende instructie wordt ingeleid door een "1" in de flip-flop te klokken. Deze "1" zal niet op $\overline{SS}$ verschijnen tenzij ALE HOOG wordt (waardoor de clear van de flip-flop wordt weggenomen). In antwoord op het HOOG gaan van $\overline{SS}$ haalt de processor een instructie op, waardoor ALE LAAG wordt. Hierdoor wordt $\overline{SS}$ via de clear-ingang gereset, zodat de processor weer stopt.

Power-down mode
De AHL-versies van de microcontrollers 8035, 8039, 8040 en de AH-versies van de 8048, 8049 en 8050 zijn voorzien van een extra schakeling die standby-bedrijf mogelijk maakt.
Hierbij wordt de voedingsspanning verwijderd van alles behalve de data-RAM, waardoor het opgenomen vermogen daalt tot 10 a 15 % van de normale waarden.

V_{CC} dient als 5 V voedingspen voor de meeste schakelingen binnen de 8048, terwijl de V_{dd}-pen alleen de RAM-array van voeding voorziet.
Onder normale omstandigheden staat op beide pennen een spanning van 5 V, terwijl in standby V_{CC} aan aarde ligt en V_{dd} op de standby-waarde wordt gehouden. Wanneer de processor door een signaal op de $\overline{RESET}$-ingang wordt gereset, wordt elke toegang tot de RAM gesperd om te verhinderen dat de inhoud van de RAM onbedoeld wordt gewijzigd.

In figuur 7/6.1-14 wordt een typisch power-down verloop getoond.

- Een dreigende fout in de voeding wordt met een door de gebruiker geïnstalleerde schakeling gedetecteerd. Het signaal hiervan moet vroeg genoeg komen om de 8048 in staat te stellen alle onmisbare data te redden voordat V_{CC} de minimale drempelwaarde overschrijdt.
- Het voedingsfout (power-fail) signaal wordt gebruikt om de processor te interrumperen en hem naar een power-fail service-subroutine te brengen.

6.1 8-bit microcontrollers van de 8048-familie

+5V
SINGLE STEP
10K
RUN
MOMENTARY PUSHBUTTON
PRESET
D
Q
$\overline{SS}$
CLOCK
CLEAR
DEBOUNCE LATCH
1/2 7400
1/2 7474
ALE

SINGLE STEP CIRCUIT

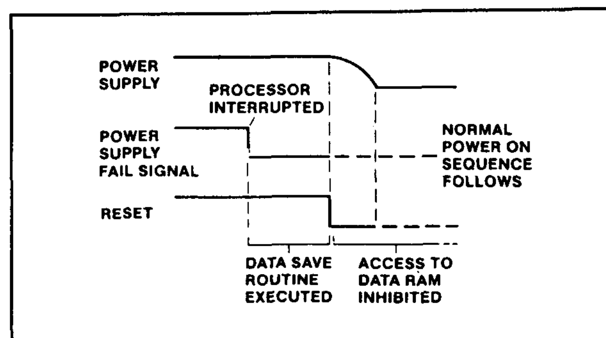
S3 | S4 | S5 | S1 | S2 | S3 | . . . . . . | S3 | S4 | S5 | S2

ALE
$\overline{SS}$
BUS
PC 0-7
P20-23
I/O
PC 8-11
I/O

SINGLE STEP TIMING

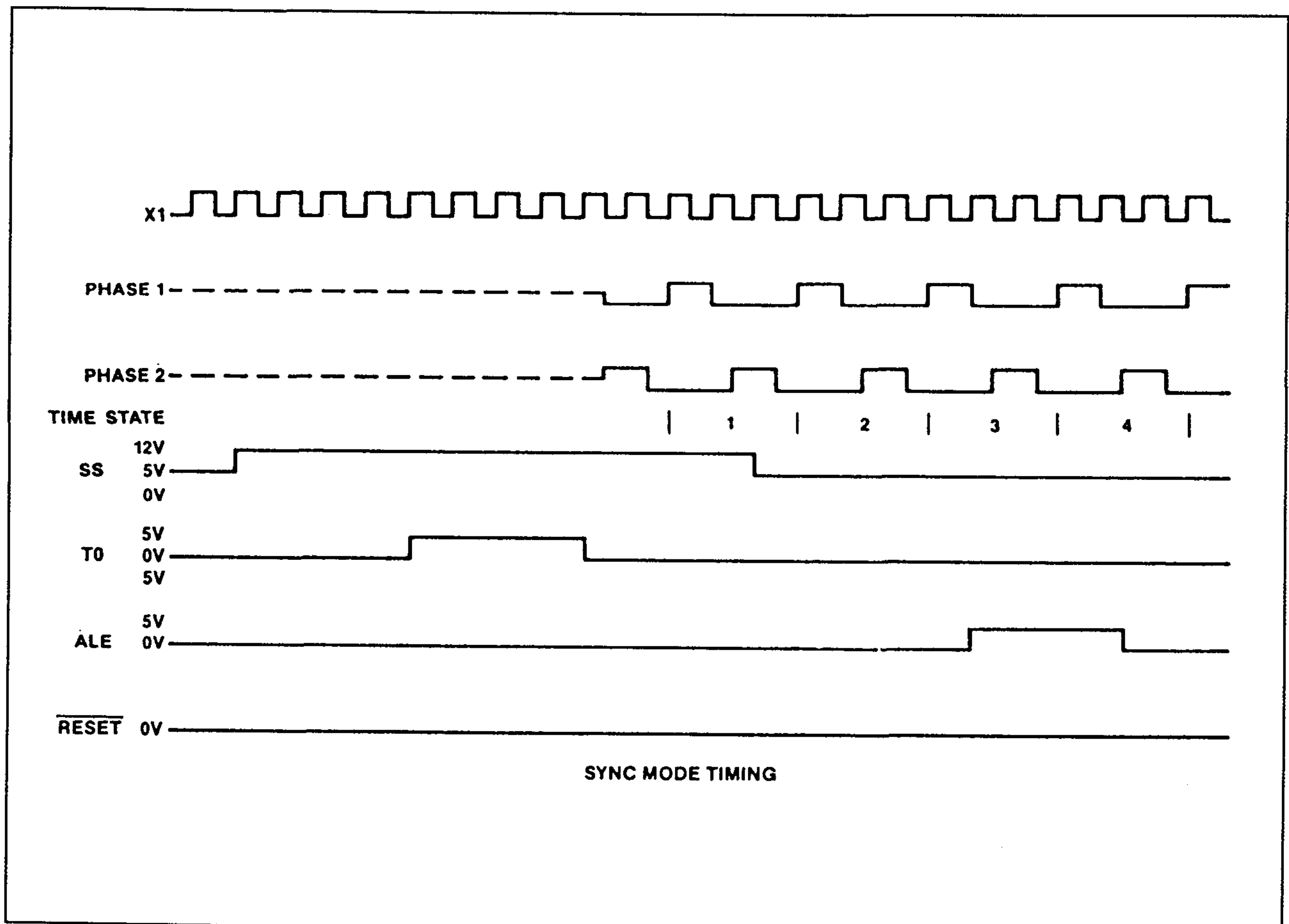
Figuur 7/6.1-13: Voorziening voor stap-voor-stap bedrijf en timing hiervan.

- Deze power-fail routine redt alle belangrijke data en machine-status door deze in de RAM-array op te slaan.
  Deze routine kan ook worden gebruikt om de V_{dd}-pen op een reserve-voeding aan te sluiten en om aan externe schakelingen te signaleren dat de power-fail routine klaar is.
- Het resetsignaal wordt gegeven om te garanderen dat geen data wordt veranderd als de voeding buiten de specificaties is. Reset moet LAAG worden gehouden totdat V_{cc} op aardniveau is.



Figuur 7/6.1-14: De handelingen bij wegvallen van de voedingsspanning.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-15: Sync mode timing.

Door resetten kan ook de vrijlooptoestand worden beëindigd. Aangezien de oscillator reeds loopt, zijn vijf machine-cyclussen voldoende om een goede werking te garanderen.

Sync mode

De 8048, 8049 en 8050 zijn uitgerust met een nieuwe SYNC mode. Deze sync mode maakt het gemakkelijker om een systeem met meerdere controllers te ontwerpen, doordat de ontwerper de schakeling in een bekende fase- en status-timing mag brengen.
De SYNC mode kan ook door automatische testapparatuur (ATE) worden gebruikt om de tester en de te testen component (device under test: DUT) snel, gemakkelijk en efficiënt op elkaar te synchroniseren.

De SYNC mode wordt mogelijk door de $\overline{SS}$-pen aan een spanning van +12 V te leggen. Om de synchronisatie te starten wordt T0 gedurende tenminste vier clockcyclussen na de verandering op $\overline{SS}$ op 5 V gebracht. Deze periode is nodig om de prescaler en de tijdgeneratoren volledig te resetten (zie figuur 7/6.1-15). T0 mag daarna op de stijgende flank van X1 weer LAAG gaan.
Twee clockcyclussen later (ook op de stijgende flank van X1) gaat de schakeling in Time State 1, Fase 1. $\overline{SS}$ gaat 4 clockpulsen later dan T0 LAAG. $\overline{RESET}$ wordt toegestaan om 5 t_{CY} (75 clockpulsen) later HOOG te gaan.

Vrijloop mode

Naast de standaard power-down mode is aan de 80C48, 80C49 en 80C50 een IDLE

mode instructie (01H) toegevoegd om de flexibiliteit te verhogen.
In de vrijloop (IDLE) mode wordt de processor bevroren, terwijl de oscillator, RAM, timer en interrupt schakelingen volledig aktief blijven.

Bij decodering van de IDL-instructie (01H) wordt de clock naar de CPU gestopt. De CPU status wordt in zijn geheel bewaard. De stackpointer, programmateller, programmastatuswoord, accumulator, RAM en alle registers handhaven hun data tijdens het vrijlopen.

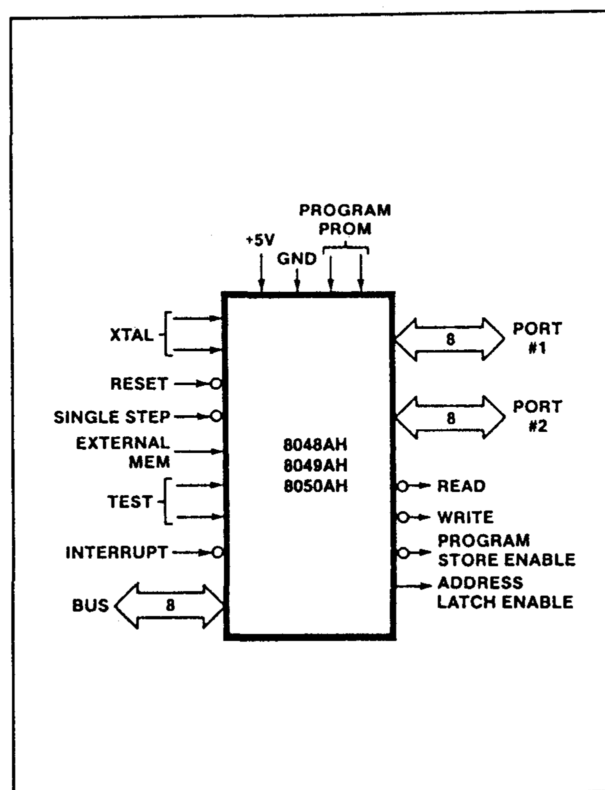
Extern gebeurt tijdens de vrijloop het volgende.

- De poorten blijven in dezelfde logische toestand als bij het begin van de vrijloop.
- Als de bus gelatcht was, blijft deze in dezelfde logische toestand staan als op het moment dat de vrijloop werd uitgevoerd.
  Als de bus zich in een hoog-impedante toestand bevond of als een extern programmageheugen wordt gebruikt blijft de bus in de zwevende toestand.
- ALE blijft in de niet-aktieve toestand (LAAG).
- $\overline{RD}$, $\overline{WR}$, PROG en $\overline{PSEN}$ blijven inaktief (HOOG).
- Indien vrijgegeven verschijnt clock op de T0-uitgang.

Er zijn drie manieren om de vrijlooptoestand op te wekken.
Door een willekeurige vrijgegeven interrupt te aktiveren (extern of door de timer) springt de CPU naar de juiste interrupt-routine. Na een RETR-instructie blijft het programma wachten bij de instructie die volgt op het adres dat de IDL-instructie bevatte. De F0 en F1 vlaggen kunnen worden gebruikt om te signaleren of de interruptie plaats vond tijdens de normale uitvoering van het programma of tijdens de vrijlooptoestand.
Dit wordt gedaan door de vlaggen te setten of te clearen voordat het vrijlopen begint. De interrupt service-routine kan de vlaggen bekijken en dienovereenkomstig handelen wanneer de vrijloop wordt beëindigd door een interrupt. Door de schakeling te resetten kan de vrijloop ook worden beëindigd. Aangezien de oscillator altijd loopt zijn vijf machinecyclussen voldoende om een goede werking te garanderen.

Beschrijving van de aansluitpennen

De microcontrollers uit de 8048-serie zijn opgenomen in 40-pens DIL-behuizingen.
In tabel 7/6.1-5 wordt een overzicht gegeven van de functies van de pennen, terwijl figuur 7/6.1-16 het logisch symbool voor de MCS-48 familie geeft.



Figuur 7/6.1-16: Logisch symbool van de 8048-familie microcontrollers.

6.1 8-bit microcontrollers van de 8048-familie

Pin Definitions and Functions

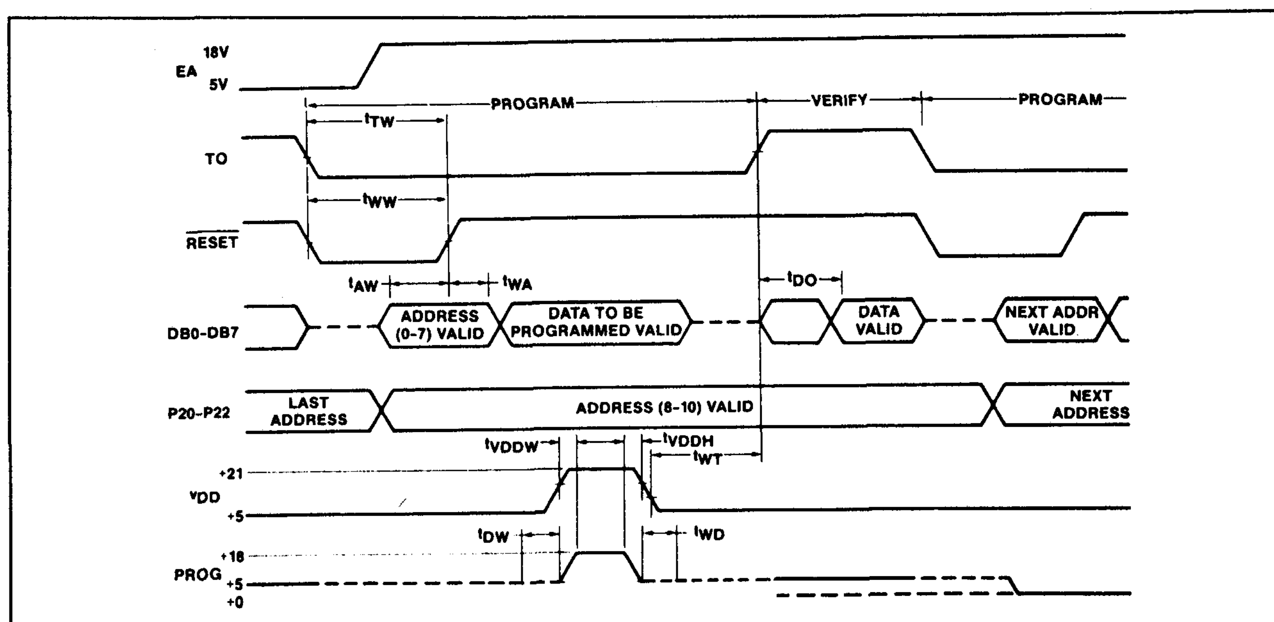
Symbol	Pin	Input (I) Output (O)	Function
T0	1	I/O	TEST PIN 1 Input pin testable using the conditional transfer instructions JT0 and JNT0. T0 can be designated as a clock output using ENT0 CLK instruction.
XTAL1, XTAL2	2,3	–	OSCILLATOR Inputs for internal oscillator with crystal or external source (non TTL VIH).
$\overline{\text{RESET}}$	4	I	$\overline{\text{RESET}}$ Input which is used to initialize the processor (active low). Also used during power down (non TTL VIH).
$\overline{\text{SS}}$	5	I	$\overline{\text{SINGLE STEP}}$ Can be used in conjunction with ALE to "single step" the processor through each instruction (active low).
$\overline{\text{INT}}$	6	I	$\overline{\text{INTERRUPT}}$ Initiates an interrupt if interrupt is enabled. Interrupt is disabled after a reset. Also testable with conditional jump instruction (active low).
EA	7	I	EXTERNAL ACCESS Input which forces all program memory fetches to reference external memory. Useful for emulation and debug, and essential for testing and program verification (active high).
$\overline{\text{RD}}$	8	O	$\overline{\text{READY}}$ Output strobe activated during a bus read. Can be used to enable data on the bus from an external device. Used as a read strobe to external data memory (active low).
$\overline{\text{PSEN}}$	9	O	$\overline{\text{PROGRAM STORE ENABLE}}$ This output occurs only during a fetch to external program memory (active low).
$\overline{\text{WR}}$	10	O	$\overline{\text{WRITE}}$ *Output strobe during a bus write* (active low). Used as a write strobe to external data memory.
ALE	11	O	ADDRESS LATCH ENABLE This signal occurs once during each cycle and is useful as a clock output. The negative edge of ALE strobes address into external data and program memory.
DB0–DB7	12-19	I/O	DATA BUS (0 TO 7) Contains the 8 low-order program counter bits during an external program memory fetch, and receives the addressed instruction under the control of $\overline{\text{PSEN}}$. Also contains the address and data during an external RAM data store instruction, under control of ALE, $\overline{\text{RD}}$, and $\overline{\text{WR}}$.
P20–P27	21-24 35-38	I/O	PARALLEL PORT (20 TO 27) 8-bit quasi-bidirectional I/O port. P20–P23 contain the four high-order program counter bits during an external program memory fetch and serve as a 4-bit I/O expander bus for SAB 8243.
PROG	25	O	PORT EXPANDER STROBE Output strobe for SAB 8243 I/O expander.
P10–P17	27-34	I/O	QUASI-BIDIRECTIONAL PORT 8-bit quasi-bidirectional I/O port.
T1	39	I	TEST PIN 1 Input pin testable using the JT1, and JTN1 instructions. Can be designated the timer/counter input using the *STRT CNT instruction.*
VCC	40		POWER SUPPLY (+5V)
VDD	26		POWER-DOWN VOLTAGE (+5V)
VSS	20		GROUND (0V)

Tabel 7/6.1-5: Beschrijving van de aansluitpennen.

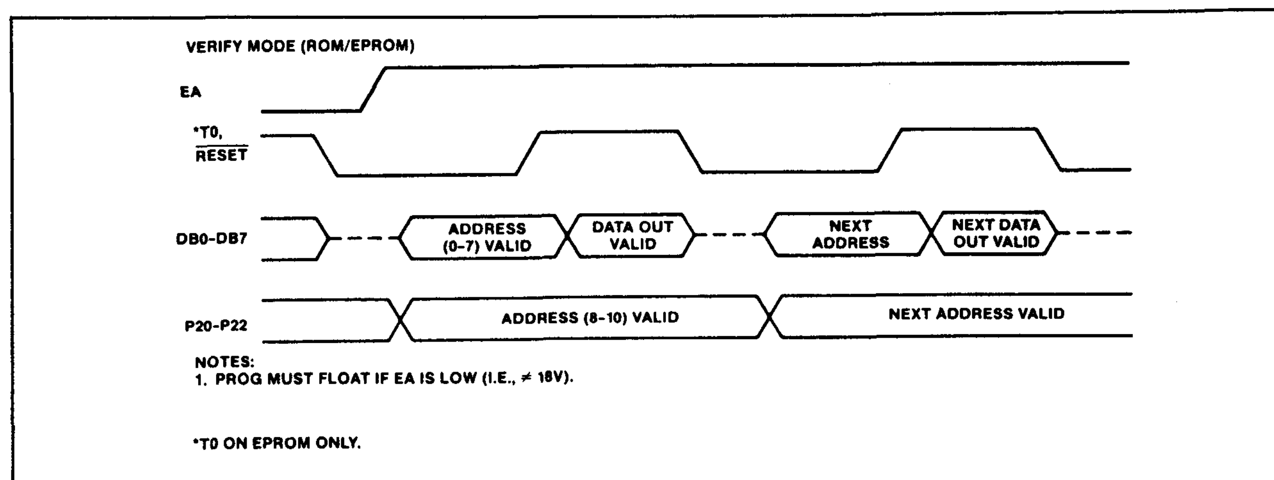
Programmeren, verifiëren en wissen van de EPROM

Het interne geheugen van de 8748 en de 8749 kan door de gebruiker zelf worden gewist en opnieuw geprogrammeerd.
Zie ook de figuren 7/6.1-17 en -18 (respectievelijk programmeren/verifiëren en alleen verifiëren) en de bij deze IC's vermelde gegevens.

Programmeren en verifiëren
In het kort komt het programmeren neer op aktiveren van de programmeer mode, een adres kiezen, het adres latchen, data aanbrengen en een programmeerpuls geven. Dit programmeer-algoritme geldt zowel voor de 8748 as de 8749. Elk woord wordt volledig geprogrammeerd en geverifiëerd voordat het volgende aan de beurt is. Voor het programmeren worden de in tabel 7/6.1-6 vermelde aansluitpennen gebruikt.



Figuur 7/6.1-17: Gecombineerd programmeren en verifiëren van de EPROM's.



Figuur 7/6.1-18: Verifiëren van ROM/EPROM.

6.1 8-bit microcontrollers van de 8048-familie

Pin	Function
XTAL 1	Clock Input (3 to 4 MHz)
Reset	Initialization and Address Latching
Test 0	Selection of Program (0V) or Verify (5V) Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-1	Address Input for 8748H
P20-2	Address Input for 8749H
V_{DD}	Programming Power Supply
PROG	Program Pulse Input
P10–P11	Tied to ground (8749H only)

Tabel 7/6.1-6: De bij het programmeren gebruikte aansluitpennen en hun functies.

Wissen van de 8748/49

Het wissen van de 8748 en de 8749 begint op te treden wanneer zij worden blootgesteld aan licht met kortere golflengten dan ongeveer 400 nm (nanometer).
Let op dat zonlicht en sommige typen fluorescerende lampen golflengten tussen 300 en 400 nm hebben. Metingen hebben uitgewezen dat de 8748 en 8749 in ongeveer 3 jaar hun gegevens verliezen wanneer zij voortdurend worden blootgesteld aan TL-licht op kamerniveau. Wanneer zij in direct zonlicht worden geplaatst duurt dit ongeveer 1 week.
Om ongewenst wissen te voorkomen moet het venster op beide typen microcontrollers dus worden afgedekt.

De aanbevolen wis-procedure voor de 8748 en de 8749 is ze bloot te stellen aan ultraviolet licht met een golflengte van 253,7 nm. De geïntegreerde dosis (= UV intensiteit X belichtingstijd) moet minimaal 15 Ws/cm^2 zijn.
Een UV-lamp zonder filter die 12 mW/cm^2 uitstraalt kan de EPROM in 15 tot 20 minuten wissen (op een afstand van circa 2,5 cm).
Na het wissen zijn alle bits LAAG (logische "0" toestand).

A	Accumulator
AC	Auxillary carry
Addr	12-bit program memory address
An	Accumulator bit n
BS	Bank switch
BUS	Bus port
CY	Carry
CLK	Clock
CNT	Event counter
Data	8-bit number or expression
DBF	Memory bank flip-flop
F0, F1	Flag 0, 1
$\overline{INT}$	Interrupt
PC	Program counter
PCn	Program counter bit n
Pp	Port 4–7 (for I/O-extension with SAB 8243)
Pr	Port 1 or port 2
PSW	Program status word
Rn	Register bit n
Rr	Register 0–7
SP	Stackpointer
T	Timer
TF	Timer flag
T0, T1	Test 0, test 1
X	Mnemonic for external RAM
#	Immediate data prefix
@	Indirect address prefix
Page	Memory block of 256 byte
( )	Content
→	is moved to
↔	is exchanged with
∧	logical AND
∨	logical OR
⊻	logical EXCLUSIVE OR
‾	Complement

Tabel 7/6.1-7: Symbolen en afkortingen die bij de instructieset worden gebruikt.

De instructieset

Inleiding

De 8048-familie heeft een uitgebreide instructieset en maakt zeer efficiënt gebruik van het programmageheugen. Alle instructies hebben een lengte van één (ruim 80 %)

of twee bytes. Bovendien worden alle instructies in één (ruim 50 %) of twee cyclussen uitgevoerd. Alle "immediate"- en I/O-instructies zijn dubbele-cyclus instructies.
Rekenkundige bewerkingen kunnen de microcomputers van de MCS-48 familie zowel binair als in BCD-formaat efficiënt uitvoeren, terwijl zij ook zeer geschikt zijn voor de enkele-bit operaties die voor controllers vereist zijn. Er zijn tevens speciale instructies aangebracht om loop-tellers, table look-up routines en N-weg branch routines te vereenvoudigen.

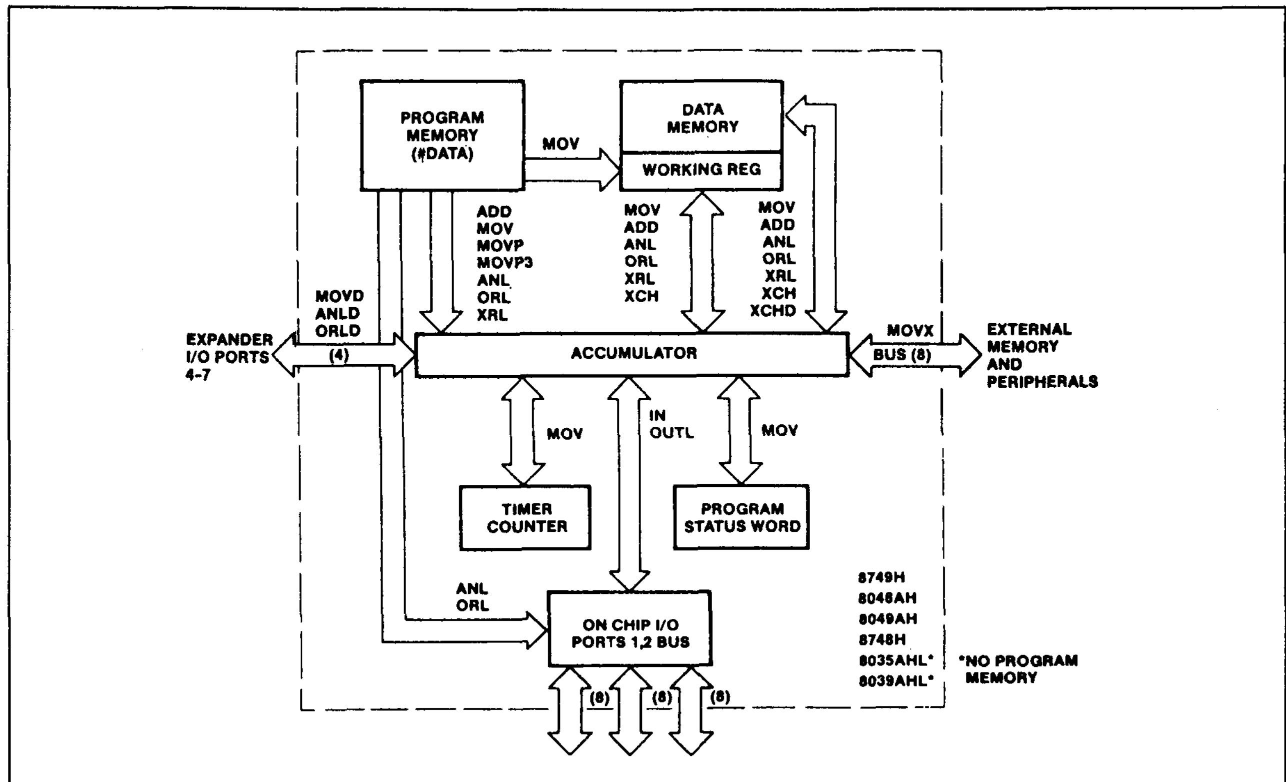
In dit gedeelte worden toelichtingen gegeven op de instructieset die over de tabellen 7/6.1-8 tot en met 7/6.1-12, opgenomen aan het einde van dit subhoofdstuk, zijn verdeeld. In tabel 7/6.1-7 zijn de gebruikte symbolen en afkortingen opgenomen.

Data-transporten
Zoals in figuur 7/6.1-19 te zien is, is de 8-bit accumulator het centrale punt voor alle datatransporten binnen de 8048. Data kan direct tussen de 8 registers van elke in bedrijf zijnde registerbank en de accu worden verplaatst doordat het bron- of bestemmingsregister door de instructie gespecificeerd wordt. De resterende lokaties in de interne RAM worden als datageheugen beschouwd en worden indirect via een in R0 of R1 opgeslagen adres geadresseerd. R0 en R1 worden ook gebruikt om eventueel aanwezig extern geheugen indirect te adresseren. Overdrachten van en naar de interne RAM vereisen één cyclus, terwijl transporten naar de externe RAM er twee nodig hebben. Constanten die in het programmageheugen zijn opgeslagen kunnen direct in de accumulator en de 8 werkregisters worden geladen. Data kan ook direct tussen de accu en de interne timer verplaatst worden of tussen de accu en het programma-statuswoord (PSW). Door in het PSW te schrijven wordt de machine-status overeenkomstig veranderd, terwijl na een interrupt de status opnieuw kan worden ingevoerd.

Accumulator operaties
De hieronder beschreven instructies die betrekking hebben op de accumulator zijn in de tabellen 7/6.1-8 en -9 opgenomen. Immediate data, data-geheugen of de werkregisters kunnen met of zonder carry worden opgeteld bij de accu (ADD).
Deze bronnen kunnen ook AND, OR of EXOR functies met de accu uitvoeren. Data kan tussen de accu en de werkregisters of het data-geheugen worden overgebracht. Beide waarden kunnen ook in één enkele operatie worden uitgewisseld.
Bovendien kunnen de laagste 4 bit van de accu worden verwisseld met de laagste 4 bit van de interne RAM lokaties. Deze instructie plus een instructie die de hoogste en laagste 4-bit helften van de accu verwisselt (SWAP) maakt eenvoudige behandeling van 4-bit aantallen (inclusief BCD-getallen) mogelijk.
Om BCD-rekenen te vereenvoudigen is een Decimal Adjust-functie toegevoegd. Deze functie dient om het resultaat van de binaire optelling van twee 2-digit BCD-getallen te corrigeren.
Door een decimale adjust op het resultaat in de accu uit te voeren wordt het gewenste BCD-resultaat bereikt.
Tenslotte kan de accu worden geïncrementeerd (telkens met 1 verhoogd), gedecrementeerd, gecleared of gecomplementeerd. Ook kan de inhoud 1 plaats linksom of rechtsom worden geroteerd, zonder carry.

Register operaties
De werkregisters kunnen via de accu worden bereikt of zij kunnen immediate met de constanten uit het programmageheugen worden geladen.
Bovendien kunnen zij worden geïncrementeerd en gedecrementeerd.
Door de "decrement en jump if not zero" instructie (DJNZ) te gebruiken kunnen zij ook dienen als loop-teller (zie ook de tabellen 7/6.1-8 en -9).
Het gehele data-geheugen, inclusief de werkregisters kan met indirecte instructies via R0 en R1 worden geadresseerd.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-19: Overzicht van de data-transporten.

Vlaggen

De 8048 heeft vier vlaggen (flags) die voor de gebruiker toegankelijk zijn: Carry, Auxiliary Carry, F0 en F1.

Met Carry wordt het overlopen van de accu aangegeven, terwijl Auxiliary Carry bij de decimal adjust operatie wordt gebruikt om een overflow tussen BCD-digits te signaleren. Beide carry-vlaggen zijn toegankelijk als deel van het programma-statuswoord en worden tijdens subroutines op de stack opgeborgen. F0 en F1 zijn algemene vlaggen die de programmeur naar eigen inzicht kan gebruiken. Beide vlaggen kunnen worden gecleared of gecomplimenteerd en getest door voorwaardelijke sprongopdrachten (zie tabel 7/6.1-12). F0 kan ook via het programma-statuswoord worden bereikt en wordt samen met de carry-vlaggen op de stack gezet.

Branch instructies

De onvoorwaardelijke sprongopdracht (jump) is twee bytes groot en maakt sprongen naar willekeurige plaatsen in de eerste 2 kB woorden van het programmageheugen mogelijk. Er kan naar de tweede 2 kB van het geheugen worden gesprongen (4 kB woorden zijn direct adresseerbaar) door eerst een select-geheugenbank instructie uit te voeren en daarna de jump-instructie (zie ook tabel 7/6.1-11). De 2 kB grens kan alleen worden overschreden via een jump of een subroutine-call instructie, hetgeen wil zeggen dat de bankswitch alleen verschijnt als de jump wordt uitgevoerd. Zodra een geheugenbank is geselecteerd zullen alle volgende jumps naar de gekozen bank plaatsvinden totdat een andere select-geheugenbank instructie is uitgevoerd. Er kan naar een subroutine in de andere bank worden gesprongen door een select-geheugenbank instructie, gevolgd door een call instructie uit te voeren. Na beëindiging van de subroutine gaat de uitvoering automatisch terug naar de oorspronkelijke bank. Let echter op dat de volgende jump weer naar de andere bank

zal gaan, tenzij de oorspronkelijke bank opnieuw geselecteerd werd.

Voorwaardelijke jumps kunnen de volgende ingangen en machine-status testen: de T0-, T1- en $\overline{\text{INT}}$-ingangspennen, accu leeg, elk bit van de accu en de carry-, F0- en F1-vlag. Voorwaardelijke sprongen maken een branch naar elk adres binnen de huidige pagina (256 woorden) mogelijk. De geteste condities zijn ogenblikkelijke waarden op het tijdstip dat de jump wordt uitgevoerd. De "jump-on-accumulator-zero" instructie (JZ) bijvoorbeeld test de accu zelf en niet een tussenkomende zero-vlag.

De "decrement register" en "jump-if-not-zero" instructies (JNZ) vormen een nuttige combinatie om een lusteller te vormen. Deze instructie kan elk van de 8 werkregisters als teller benoemen en maakt sprongen binnen de huidige werkpagina mogelijk.

Een single-byte indirecte jump instructie maakt het mogelijk het programma te richten op verschillende lokaties die gebaseerd zijn op de inhoud van de accu. De inhoud van de accu wijst naar een plaats in het programmageheugen die het sprongadres bevat. Het 8-bit sprongadres heeft betrekking op de huidige werkpagina. Deze instructie kan bijvoorbeeld worden gebruikt om naar verschillende, op in de accu aanwezige ASCII-karakters gebaseerde, routines te springen. Op deze wijze kunnen ASCII-toetsen worden gebruikt om verschillende routines in te leiden.

Subroutines
Subroutines worden bereikt door het uitvoeren van een "call" instructie. Calls kunnen de vorm hebben van onvoorwaardelijke jumps naar elk adres binnen een bank van 2 kB woorden, terwijl sprongen over de 2 kB grens op dezelfde manier worden uitgevoerd. Er zijn twee aparte return instructies die bepalen of de status na terugkeer uit de subroutine al dan niet opnieuw wordt opgeslagen (zie tabel 7/6.1-10).

De return en restore status instructie (RETR) geeft ook het einde van een eventuele aan de gang zijnde interrupt service-routine aan.

Timer instructies
De interne 8-bit timer/counter kan via de accu worden geladen en uitgelezen. De teller kan hierbij gestopt zijn of doortellen. De teller kan als timer met een interne clock worden gebruikt of als een timer of teller met een externe clock die op de T1-pen wordt aangesloten. De instructie die wordt uitgevoerd bepaalt welke clock wordt gebruikt (zie tabel 7/6.1-12). Er is één instructie voor het stoppen van de teller, of die nu met een interne of een externe clock werkt. Bovendien kan de timer-interrupt worden vrijgegeven of gesperd.

Control instructies
Er zijn twee instructies die de externe interrupties kunnen vrijgeven of sperren (tabel 7/6.1-12). Interrupts zijn direct na het aanzetten gesperd en worden tijdens het uitvoeren van een interrupt service routine automatisch gesperd en daarna weer vrijgegeven.
De 8048 heeft vier instructies om geheugenbanken te selecteren: twee om de aktieve registerbank aan te wijzen en twee om de geheugenbanken voor het programma te besturen.
De omschakelinstructie voor de werkende registerbank stelt de programmeur in staat de gebruikte 8-registerbank onmiddellijk te vervangen door een tweede. Hierdoor komen feitelijk 16 werkregisters ter beschikking of het geeft de mogelijkheid om snel de inhoud van de registers te bewaren bij een interrupt. De gebruiker staat het vrij om bij interrupts wel of niet van bank te wisselen. Wanneer echter van bank wordt gewisseld, zal na de return automatisch de originele bank terugkeren.
Een speciale instructie zet een interne clock (de XTAL frequentie, gedeeld door 3) op uitgang T0. Deze clock kan dan in het systeem worden toegepast.

6.1 8-bit microcontrollers van de 8048-familie

Input/Output instructies

De poorten 1 en 2 zijn 8-bit I/O-poorten die van en naar de accumulator kunnen worden geladen. Uitgangen worden statisch gelatcht, maar ingangen niet (en dienen dus te worden gelezen met aanwezige signalen). Bovendien kunnen AND- en OR-operaties direct worden uitgevoerd met immediate data uit het programmageheugen en poort 1 en poort 2, waarbij het resultaat in de poort achterblijft.

Dit maakt het mogelijk om met "maskeringen" uit het programmageheugen individuele bits van de I/O-poorten te setten of te resetten (zie tabel 7/6.1-10).

De pennen van de poorten kunnen als ingang worden geconfigureerd door eerst een "1" naar de betreffende pen te schrijven.

Een 8-bit poort die BUS wordt genoemd kan ook via de accu worden bereikt. Ook hierop kunnen AND- en OR-handelingen worden verricht. In tegenstelling tot de poorten 1 en 2 moeten alle 8 lijnen van BUS echter tegelijk als ingang of als uitgang worden behandeld. Behalve als statische poort kan BUS ook worden gebruikt als een echte synchrone bidirectionele poort. Dit wordt gedaan door gebruik te maken van de Move External instructies. Wanneer deze instructies worden uitgevoerd, wordt een overeenkomstige $\overline{\text{READ}}$ of $\overline{\text{WRITE}}$ puls gegenereerd en is data op dat moment beschikbaar.

Als geen data wordt overgebracht, verkeert BUS in de hoog-impedante toestand. Let op dat de OUTL, ANL en ORL instructies alleen met intern programmageheugen kunnen worden gebruikt.

6.1 8-bit microcontrollers van de 8048-familie

Instruction Set Summary (cont'd)

Mnemonic	Function	Description	Hex Code	Flag	Byte	Cycle
Arithmetic accumulator instructions						
ADD A, Rr	(A)+(Rr) → A	Add register contents to accumulator	68–6F	AC, CY	1	1
ADD A, @ Rr	(A)+((Rr)) → A	Add data memory contents to accumulator	60 61	AC, CY	1	1
ADD A, # data	(A)+data → A	Add immediate data to accumulator	03	AC, CY	2	2
ADDC A, Rr	(A)+(Rr)+(CY) → A	Add carry and register contents to accumulator	78–7F	AC, CY	1	1
ADDC A, @ Rr	(A)+((Rr))+(CY) → A	Add carry and data memory contents to accumulator	70 71	AC, CY	1	1
ADDC A, # data	(A)+data+(CY) → A	Add carry and immediate data to accumulator	13	AC, CY	2	2
INC A	(A)+1 → A	Increment accumulator	17		1	1
DEC A	(A)−1 → A	Decrement accumulator	07		1	1
DA A		Decimal adjust accumulator	57	AC, CY	1	1
Arithmetic register instructions						
INC Rr	(Rr)+1 → Rr	Increment register	18–1F		1	1
DEC Rr	(Rr)−1 → Rr	Decrement register	C8–CF		1	1
INC @ Rr	((Rr))+1 → (Rr)	Increment data memory location	10–11		1	1
DJNZ Rr, addr	(Rr)−1 → Rr if (Rr) ≠ 0 Adr → PC0–7	Decrement register and test register if zero	E8–EF		2	2
Logical accumulator and register instructions						
ANL A, Rr	(A) ∧ (Rr) → A	Logical AND accumulator with register mask	58–5F		1	1
ANL A, @ Rr	(A) ∧ ((Rr)) → A	Logical AND accumulator with memory mask	50 51		1	1
ANL A, # data	(A) ∧ data → A	Logical AND accumulator with immediate mask	53		2	2
ORL A, Rr	(A) ∨ (Rr) → A	Logical OR accumulator with register mask	48–4F		1	1
ORL A, @ Rr	(A) ∨ ((Rr)) → A	Logical OR accumulator with memory mask	40 41		1	1
ORL A, # data	(A) ∨ data → A	Logical OR accumulator with immediate mask	43		2	2
XRL A, Rr	(A) ⊻ (Rr) → A	Logical XOR accumulator with register mask	D8–DF		1	1
XRL A, @ Rr	(A) ⊻ ((Rr)) → A	Logical XOR accumulator with memory mask	D0 D1		1	1
XRL A, # data	(A) ⊻ data → A	Logical XOR accumulator with immediate mask	D3		2	2
CLR A	0 → A	Clear accumulator	27		1	1
CPL A	$(\bar{A})$ → A	Complement accumulator	37		1	1

Tabel 7/6.1-8: Instructieset (rekenkundige en logische operaties met de accumulator en registers).

6.1 8-bit microcontrollers van de 8048-familie

Instruction Set Summary

Mnemonic	Function	Description	Hex Code	Flag	Byte	Cycle
Accumulator and register move instructions						
MOV A, Rr	(Rr) → A	Move register contents	F8–FF		1	1
MOV A, @ Rr	((Rr)) → A	Move data memory contents to accumulator	F0–F1		1	1
MOV A, # data	Data → A	Move immediate data data to accumulator	23		2	2
MOV A, PSW	(PSW) → A	Move PSW contents to accumulator	C7		1	1
MOV PSW, A	(A) → PSW	Move accumulator contents to PSW	D7	CY, AC	1	1
MOV Rr, A	(A) → Rr	Move accumulator contents to register	A8–AF		1	1
MOV @ Rr, A	(A) → (Rr)	Move accumulator contents to data memory	A0–A1		1	1
MOV Rr, # data	Data → Rr	Move immediate data to register	B8–BF		2	2
MOV @ Rr, # data	Data → (Rr)	Move immediate data to data memory	B0–B1		2	2
MOVX A, @ Rr	((Rr)) → A	Move external data memory contents to accumulator	80–81		1	2
MOVX @ Rr, A	(A) → (Rr)	Move accumulator contents to external data memory	90–91		1	2
XCH A, Rr	(Rr) ↔ (A)	Exchange accumulator and register contents	28–2F		1	1
XCH A, @ Rr	((Rr)) ↔ (A)	Exchange accumulator and data memory contents	20–21		1	1
XCHD A, @ Rr	((Rr))0–3 ↔ (A)0–3	Exchange accumulator and data memory 4-bit data	30–31		1	1
MOVP3 A, @ A	(PC) save (A) → PC 0–7 011 → PC 8–11 ((PC)) → A PC restor	Move page 3 data to accumulator	E3		1	2
MOVP A, @ A	(PC) save (A) → PC 0–7 ((PC)) → A PC restor	Move current page data to accumulator	A3		1	2
SWAP A	(A) 0–3 ↔ (A) 4–7	Swap nibble within accumulator	47		1	1

Tabel 7/6.1-9: Instructieset (move instructies voor accumulator en registers).

6.1 8-bit microcontrollers van de 8048-familie

Instruction Set Summary (cont'd)

Mnemonic	Function	Description	Hex Code	Flag	Byte	Cycle
Port move instructions						
IN A, Pr	(Pr) → A	Input port 1 or 2 data to accumulator	09–0A		1	2
OUTL Pr, A	(A) → Pr	Output accumulator data to port 1 or 2	39–3A		1	2
ANL Pr, # data	(Pr) ∧ data → Pr	Logical AND port 1–2 with immediate mask	99–9A		2	2
ORL Pr, # data	(Pr) ∨ data → Pr	Logical OR port 1–2 with immediate mask	89–8A		2	2
INS A, bus	(bus) → A	Strobed input of bus data to accumulator	08		1	2
OUTL bus, A	(A) → bus	Output accumulator data to bus	02		1	2
ANL bus, # data	(bus) ∧ data → bus	Logical AND bus with immediate mask	98		2	2
ORL bus, # data	(bus) ∨ data → bus	Logical OR bus with immediate mask	88		2	2
MOVD A, PP	(PP) → A 0–3 0 → A 4–7	Move port 4–7 of SAB 8243 to accumulator — Port 4 5 6 7	0C 0D 0E 0F		1 1 1 1	2 2 2 2
MOVD PP, A	(A)0–3 → PP	Move accumulator to port 4–7 of SAB 8243 — Port 4 5 6 7	3C 3D 3E 3F		1 1 1 1	2 2 2 2
ANLD PP, A	(A)0–3 ∧ (PP) → PP	Logical AND port 4–7 of SAB 8243 with accumulator mask — Port 4 5 6 7	9C 9D 9E 9F		1 1 1 1	2 2 2 2
ORLD PP, A	(A)0–3 ∨ (PP) → PP	Logical OR port 4–7 of SAB 8243 with accumulator mask — Port 4 5 6 7	8C 8D 8E 8F		1 1 1 1	2 2 2 2
Subroutine instructions						
CALL addr	(PC0–11,PSW) → (SP) (SP)+1 → SP addr0–7 → PC0–7 addr8–10 → PC8–10 DBF → PC11	Subroutine call — Page 0 1 2 3 4 5 6 7	14 34 54 74 94 B4 D4 F4		2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2
RET	(SP)–1 → SP ((SP)) → PC	Return without PSW restore	83		1	2
RETR	(SP)–1 → SP ((SP)) → PC ((SP)) → PSW4–7	Return with PSW restore	93	AC, CY	1	2

Tabel 7/6.1-10: Instructieset (move instructies voor de poorten en subroutine-commando's).

6.1 8-bit microcontrollers van de 8048-familie

Instruction Set Summary (cont'd)

Mnemonic	Function	Description	Hex Code	Flag	Byte	Cycle
Branch instructions						
JMP addr	addr0−7 → PC0−7 addr8−10 → PC8−10 DBF → PC11	Direct jump within 2K-block Page 0 1 2 3 4 5 6 7	04 24 44 64 84 A4 C4 E4		2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2
JMPP @A	((A)) → PC0−7	Indirect jump within page	B3		1	2
JC addr	If (CY) = 1 addr → PC0−7	Jump if carry is set	F6		2	2
JNC addr	If (CY) = 0 addr → PC0−7	Jump if carry is not set	E6		2	2
JZ addr	If (A) = 0 addr → PC0−7	Jump if accumulator is zero	C6		2	2
JNZ addr	If (A) ≠ 0 addr → PC0−7	Jump if accumulator is not zero	96		2	2
JT0 addr	If T0 = 1 addr → PC0−7	Jump if test 0 is high	36		2	2
JNT0 addr	If T0 = 0 addr → PC0−7	Jump if test 0 is low	26		2	2
JT1 addr	If T1 = 1 addr → PC0−7	Jump if test 1 is high	56		2	2
JNT1 addr	If T1 = 0 addr → PC0−7	Jump if test 1 is low	46		2	2
JF0 addr	If F0 = 1 addr → PC0−7	Jump if flag 0 is set	B6		2	2
JF1 addr	If F1 = 1 addr → PC0−7	Jump if flag 1 is set	76		2	2
JTF addr	If TF = 1 addr → PC0−7 0 → TF	Jump if timer flag is set	16	TF	2	2
JNI addr	If $\overline{INIT}$ = 0 addr → PC0−7	Jump if interrupt input is low	86		2	2
JBn addr	If bit n = 1 addr → PC0−7	Jump if accumulator bit n is set n = 0 1 2 3 4 5 6 7	12 32 52 72 92 B2 D2 F2		2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2

Tabel 7/6.1-11: Instructieset (branch instructies).

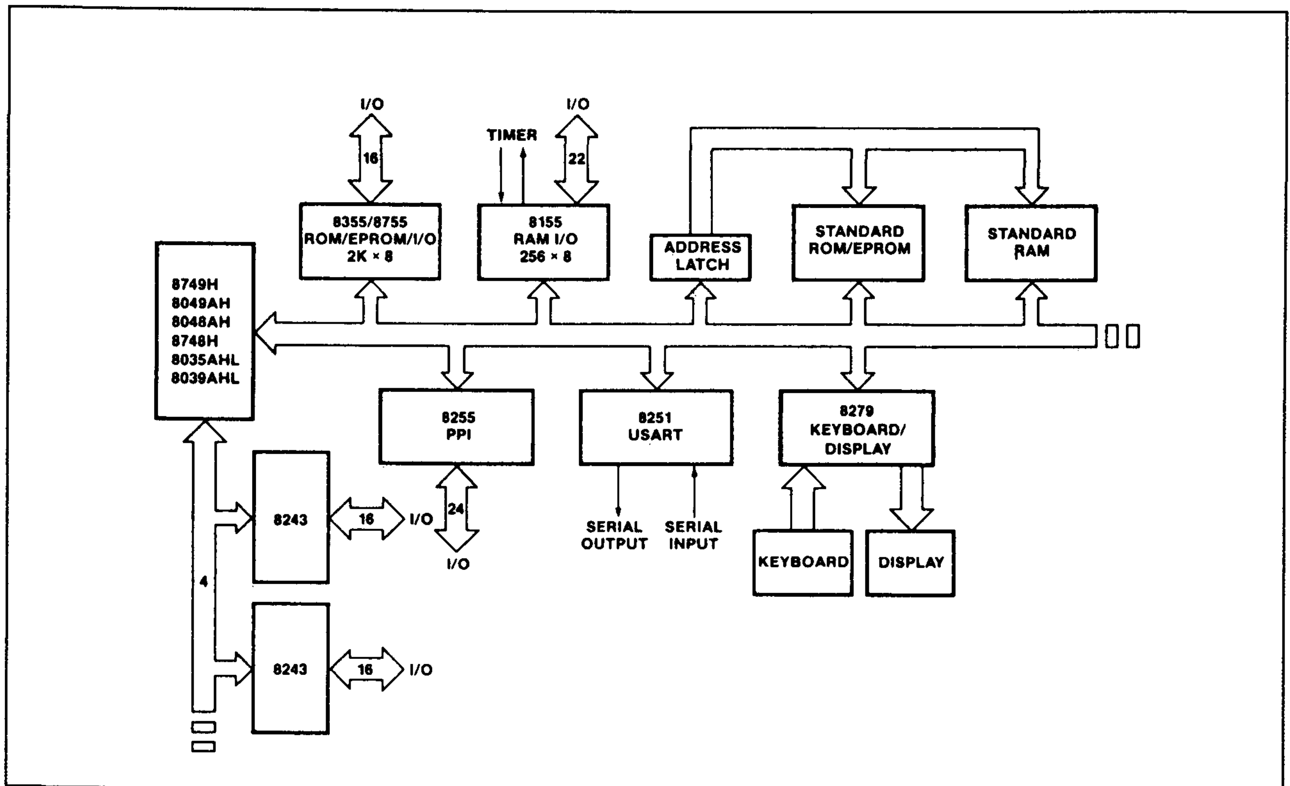
6.1 8-bit microcontrollers van de 8048-familie

Instruction Set Summary (cont'd)

Mnemonic	Function	Description	Hex Code	Flag	Byte	Cycle
Control instructions						
STRT T		Start timer	55		1	1
STRT CNT		Start event counter	45		1	1
STOP TCNT		Stop timer/event-counter	65		1	1
EN TCNTI		Enable timer/counter interrupt	25		1	1
DIS TCNTI		Disable timer/counter interrupt	35		1	1
EN I		Enable external interrupt	05		1	1
DIS I		Disable external interrupt	15		1	1
SEL RB0	0 → BS	Select register bank 0	C5		1	1
SEL RB1	1 → BS	Select register bank 1	D5		1	1
SEL MB0	0 → DBF	Select memory bank 0	E5		1	1
SEL MB1	1 → DBF	Select memory bank 1	F5		1	1
ENT0 CLK		Enable clock output	75		1	1
NOP		The NOP instruction	00		1	1
Timer/counter move instructions						
MOV A, T	(T) → A	Move timer/counter contents to accumulator	42		1	1
MOV T, A	(A) → T	Move accumulator contents to timer/counter	62		1	1
Rotate instructions						
RL A	(An) → An+1	Rotate accumulator left without carry	E7		1	1
RLC A	(An) → An+1 (A7 → CY (CY) → A0	Rotate accumulator left through carry	F7	CY	1	1
RR A	(An+1) → An	Rotate accumulator right without carry	77		1	1
RRC A	(An+1) → An (A0) → CY (CY) → A7	Rotate accumulator right through carry	67	CY	1	1
Flag instructions						
CLR C	0 → CY	Clear carry bit	97	CY	1	1
CPL C	$(\overline{CY})$ → CY	Complement carry bit	A7	CY	1	1
CLR F0	0 → F0	Clear flag 0	85		1	1
CPL F0	$(\overline{F0})$ → F0	Complement flag 0	95		1	1
CLR F1	0 → F1	Clear flag 1	A5		1	1
CPL F1	$(\overline{F1})$ → F1	Complement flag 1	B5		1	1

Tabel 7/6.1-12: Instructieset (besturings, timer/counter move, roteer en vlag instructies).

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-20: Uitbreidingsmogelijkheden van de 8048-familie.

Uitbreiding 8048-systemen

Inleiding

Indien de mogelijkheden van een single-chip microcontroller uit de 8048-familie onvoldoende zijn, maken de beschikbare signalen uitbreidingen tot een systeem met verschillende soorten externe geheugen-, I/O- of speciale schakelingen mogelijk. Door bank-switching technieken toe te passen is de maximale capaciteit praktisch onbegrensd. De processor kan eenvoudig worden uitgebreid met:

- Programmageheugen tot 4 kB;
- Datageheugen tot 320 woorden (384 woorden met de 8049);
- Een onbeperkt aantal in- en uitgangsschakelingen;
- Speciale functies door gebruik te maken van periferie-schakelingen van de 8080/8085-familie.

Uitbreiding kan op twee manieren worden verkregen (zie ook figuur 7/6.1-20).

- Expander I/O
  Door gebruik te maken van een speciale I/O Expander-schakeling, de 8243, komen vier extra 4-bit Input/Output poorten ter beschikking, waarbij alleen de laagste helft (4 bit) van poort 2 wordt opgeofferd aan de inter-IC communicatie. Op deze 4-bit bus kunnen meerdere 8243's worden aangesloten als wordt voorzien in "chip select" lijnen.

- Standaard 8085 Bus
  Een poort van de 8048/8049 komt overeen met de 8-bit bidirectionele databus van de 8085 microcomputer, waardoor deze op talloze standaard geheugens en periferie-schakelingen van de 8080/8085 kan worden aangesloten.

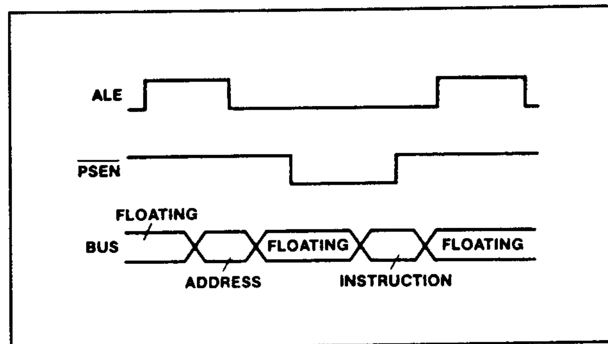
Uitbreiding van het programmageheugen

Het programmageheugen kan worden uitgebreid door de 8085 BUS voorziening van de MCS-48 toe te passen.
Alle uitlezingen van het programmageheugen op adressen die lager zijn dan 1024 voor de 8048 of 2048 voor de 8049 gebeuren inwendig.
Behalve de altijd aanwezige ALE worden hierbij geen externe signalen opgewekt. Vanaf adres 1024 in de 8048 initieert de processor automatisch externe programmageheugen "fetches" (= ophalen).

Externe Instructie Fetch cyclus

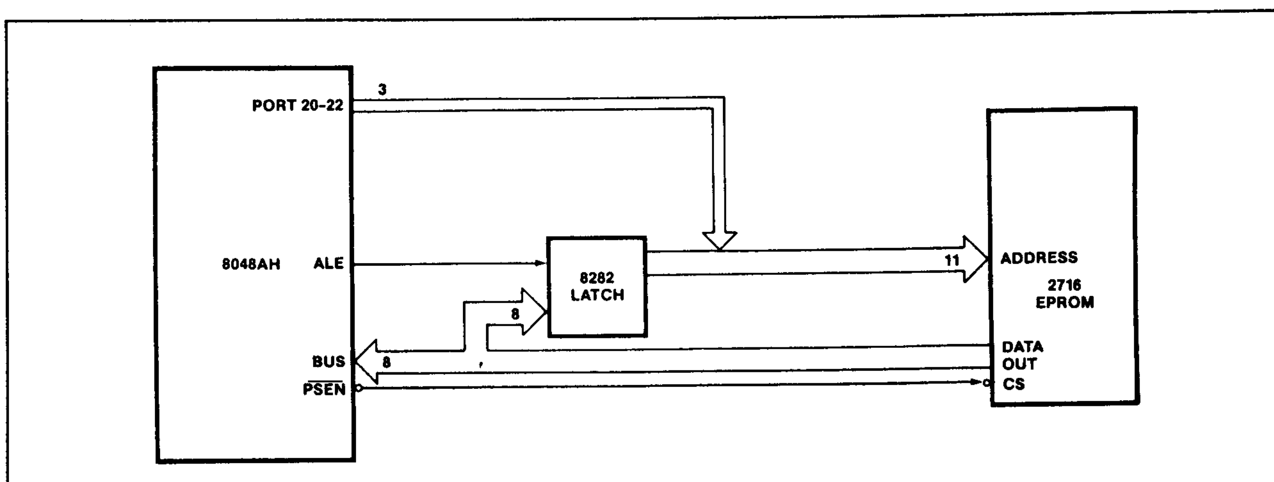
Zoals ook in figuur 7/6.1-21 te zien is, gebeurt voor alle instructie-fetches vanaf adres 1024 (of 2048) het volgende.

- De inhoud van de 12-bit programmateller wordt op BUS en de onderste helft van poort 2 gezet.
- Address Latch Enable (ALE) geeft de tijd aan waarin het adres geldig is. De achterflank van ALE wordt gebruikt om het adres extern te latchen.
- Program Store Enable ($\overline{\text{PSEN}}$) signaleert dat de processor bezig is met het ophalen van een externe instructie en dient om het externe geheugen vrij te geven.
- BUS keert terug in de ingangsmode (zwevend) en de processor accepteert de 8-bit inhoud als instructiewoord.



Figuur 7/6.1-21: Timing bij het ophalen van een instructie uit het externe programmageheugen.

In figuur 7/6.1-22 is een praktische schakeling gegeven. Het programmageheugen is hierin met 2 kB uitgebreid met een EPROM van het type 2716. In dit geval is geen chipselect decodering nodig en wordt $\overline{\text{PSEN}}$ direct op de CS-ingang van de 2716 aangesloten. Indien slechts 2 kB programmageheugen nodig is kan dezelfde opstelling worden toegepast, waarbij de 8048 dan vervangen wordt door een 8035. Door in dezelfde schakeling een 8049 te gebruiken, komt 4 kB programmageheugen ter beschikking.



Figuur 7/6.1-22: Uitbreiding van het programmageheugen door middel van standaard geheugenprodukten.

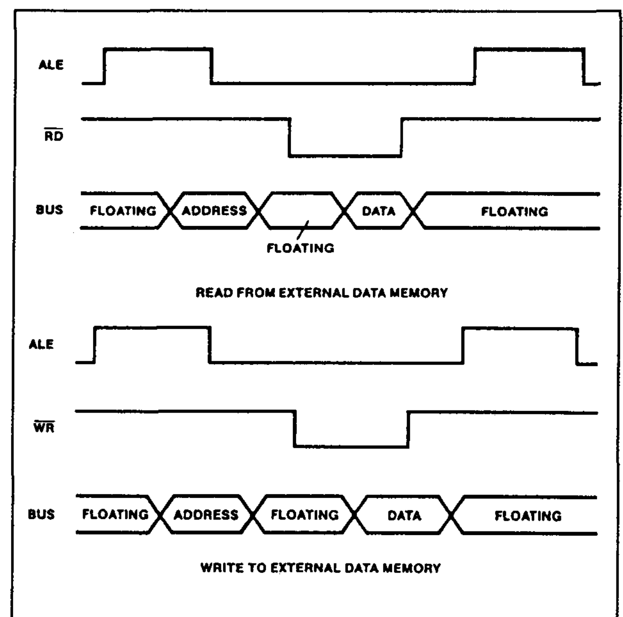
6.1 8-bit microcontrollers van de 8048-familie

Alle instructie-fetches, inclusief die naar interne adressen, kunnen overigens extern worden gemaakt door de EA-pen van de 8048/49/50 te aktiveren. De processoren 8035/39/40, die geen programmageheugen hebben, werken altijd in de externe mode (EA = 5 V).

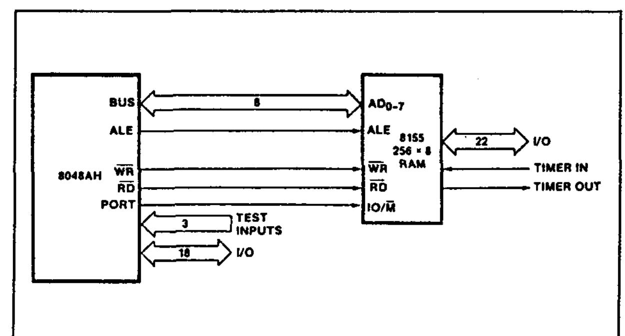
Uitbreiding van het datageheugen
Het datageheugen kan (verder dan de 64 woorden in een 8048) worden uitgebreid door gebruik te maken van de 8085-bus voorziening.

- Lees/schrijf cyclus
  Alle adressen en data worden over de 8 lijnen van BUS getransporteerd.
  Zoals figuur 7/6.1-23 ook laat zien, vindt bij een lees- of schrijfcyclus het volgende plaats.
  - De inhoud van de registers R0 en R1 wordt op BUS gezet.
  - ALE geeft aan dat het adres geldig is. De dalende flank van ALE wordt gebruikt om het adres extern te latchen.
  - Met een lees-($\overline{RD}$) of schrijfpuls ($\overline{WR}$) op de overeenkomstige pennen van de 8048 wordt het type geheugentoegang aangegeven. Uitgangsdata is geldig op de achterflank van $\overline{WR}$ en ingangsdata moet stabiel zijn op de achterflank van $\overline{RD}$.
  - Data (8 bit) wordt via BUS gelezen of geschreven.

- Adresseren van extern datageheugen
  Toegang tot extern datageheugen kan worden verkregen met de twee-cyclus move instructies MOVXA, @R en MOVX@R, A.
  Met deze instructies wordt 8-bit data verplaatst tussen de accumulator en de externe geheugenplaats die door de inhoud van de RAM-pointerregisters R0 en R1 wordt aangewezen. Hierdoor kunnen naast de residente lokaties nog 256 extra plaatsen worden geadresseerd. Extra pagina's kunnen worden toegevoegd door "bank-omschakeling" met behulp van extra uitgangslijnen van de 8048. In figuur 7/6.1-24 is te zien hoe de 8048 bijvoorbeeld kan worden uitgebreid met een geheugen- en I/O-expansie IC van het type 8155.
  Aangezien de 8155 een interne 8-bit adreslatch heeft, kan hij direct (zonder externe latch) op de 8048 worden aangesloten. De 8155 voorziet in 256 extra RAMwoorden, alsmede 22 in-/uitgangslijnen en een 14-bit timer.

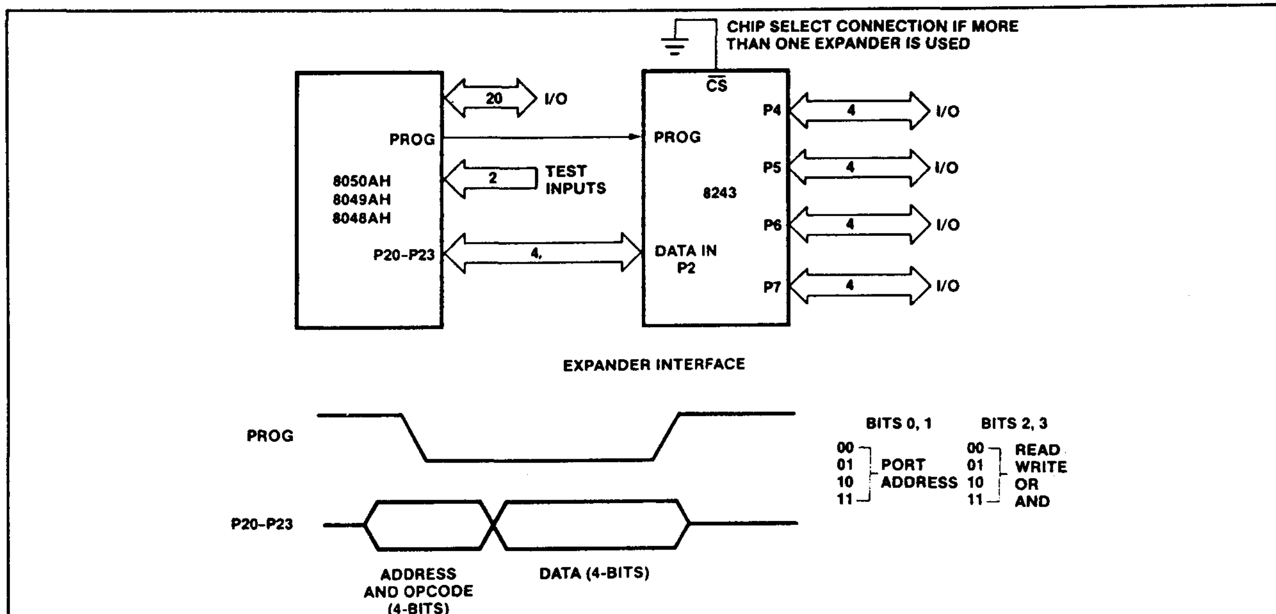


Figuur 7/6.1-23: Timing van externe datageheugen-toegangen.



Figuur 7/6.1-24: Uitbreiding van de 8048 met een 8155 (256 x 8 RAM + timer + 22 I/O-lijnen).

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-25: Aansluiting van een 8343 I/O-expander op de 8048.

Uitbreiding van de in-/uitgangen

Het aantal in- en uitgangen van de 8048 kan op vier manieren worden vergroot, door toepassing van:

- de 8243, een speciale goedkope expander;
- standaard I/O-schakelingen voor 8080/8085 microcomputers;
- gemengde geheugen-I/O-schakelingen, zoals de 8155, 8355 en 8755;
- standaard TTL-schakelingen.

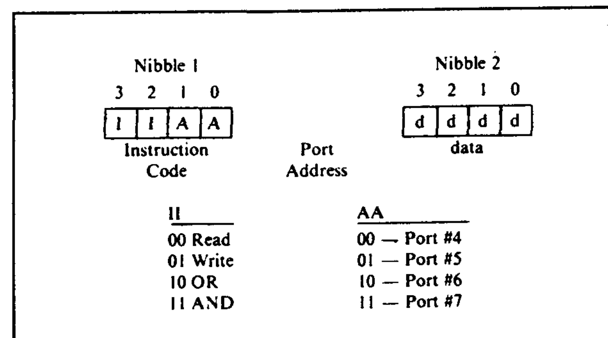
Voor kleine systemen kan de I/O-expander 8243 het best worden toegepast. Zoals ook in figuur 7/6.1-25 wordt getoond, heeft deze schakeling slechts 4 poortlijnen (de laagste helft van poort 2) nodig om met de 8048 te communiceren. De 8243 bevat vier 4-bit in-/uitgangspoorten die als uitbreiding van de in de 8048 aanwezige I/O-poorten dienen en geadresseerd worden als poort 4 tot en met 7.

Met deze poorten zijn de volgende operaties mogelijk:

- Datatransport van accumulator naar poort en omgekeerd;
- AND- of OR-functie van accumulator op poort.

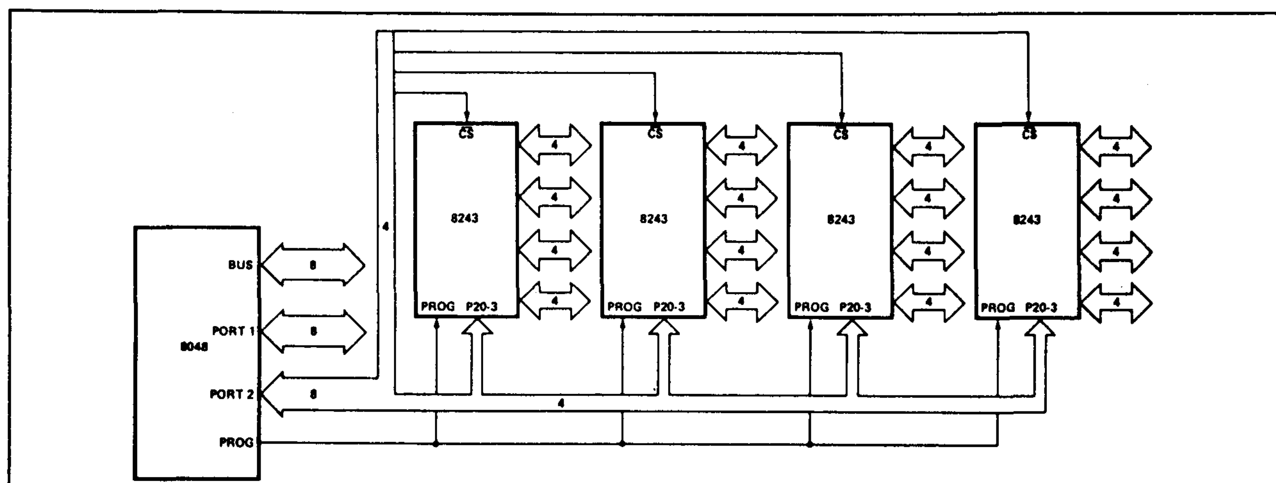
Een 4-bit overdracht van een poort naar de laagste helft van de accumulator zet de belangrijkste 4 bit op nul. Alle communicaties tussen de 8048 en de 8243 gaan via de laagste helft van poort 2 (P20 tot en met P23), waarbij de timing afkomstig is van een uitgangspuls op de PROG-pen van de processor. Elke overdracht bestaat uit twee 4-bit nibbles. De eerste bevat de opcode en het poortadres en de tweede de eigenlijke 4-bit data (zie figuur 7/6.1-26).

Een HOOG-naar-LAAG overgang op de PROG-lijn geeft aan dat een adres aanwezig is, terwijl een LAAG-naar-HOOG overgang de aanwezigheid van data signaleert.



Figuur 7/6.1-26: Inhoud van de nibbles bij gebruik van een 8243.

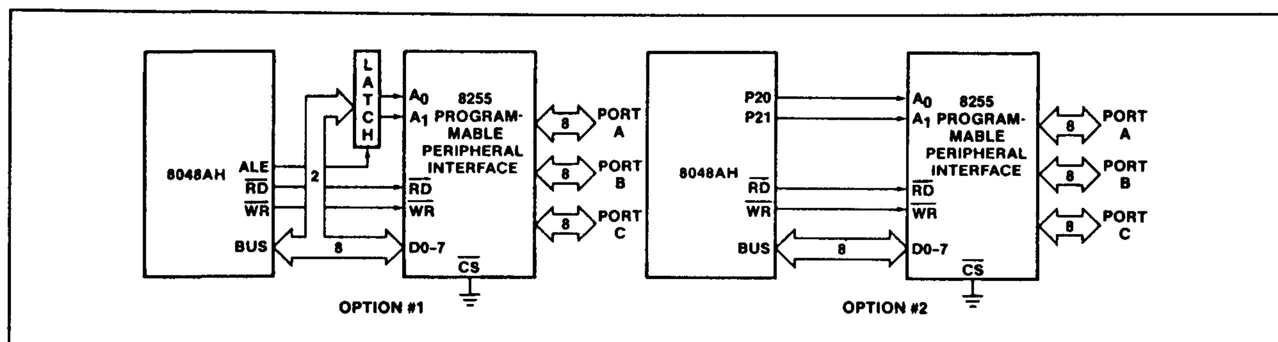
6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-27: Uitbreiding van een 8048-systeem met meerdere 8243's.

Uiteraard kunnen op de 4-bit bus van figuur 7/6.1-25 meerdere 8243's worden aangesloten. Voor de chip-selects zijn dan extra uitgangslijnen van de 8048 nodig (zie figuur 7/6.1-27).

In figuur 7/6.1-28 zijn twee manieren te zien waarop de 8048 met een standaard interface-IC, de Programmable Peripheral Interface 8255, kan worden verbonden, waardoor drie 8-bit bidirectionele poorten ontstaan.



Figuur 7/6.1-28: Twee voorbeelden van mogelijke aansluitingen van de 8048 microcontroller op de PPI 8255.

Device	Internal Memory		RAM Standby
8050AH	4K × 8 ROM	256 × 8 RAM	yes
8049AH	2K × 8 ROM	128 × 8 RAM	yes
8048AH	1K × 8 ROM	64 × 8 RAM	yes
8040AHL	none	256 × 8 RAM	yes
8039AHL	none	128 × 8 RAM	yes
8035AHL	none	64 × 8 RAM	yes

Tabel 7/6.1-13: Overzicht van de verschillen tussen de zes typen microcontrollers.

8048, 8049, 8050 (met ROM) 8035, 8039, 8040 (zonder ROM)

NMOS microcontroller (single-chip 8-bit microprocessor)

De leden van deze familie NMOS microcontrollers kunnen volledig zelfstandig werken, hebben 27 in-/uitgangslijnen, een 8-bit timer/counter en oscillator/clock schakelingen aan boord.

In de figuren 7/6.1-29, -30 en -31 zijn respectievelijk het blokschema, het logisch symbool en de aansluitingen van van de vermelde typen te zien. Tabel 7/6.1-13 geeft de verschillen tussen de zes typen weer.

De beschrijving van de aansluitpennen en de benodigde signalen zijn vermeld in tabel 7/6.1-5, terwijl de instructieset reeds in de tabellen 7/6.1-7 tot en met 7/6.1-12 werd behandeld.

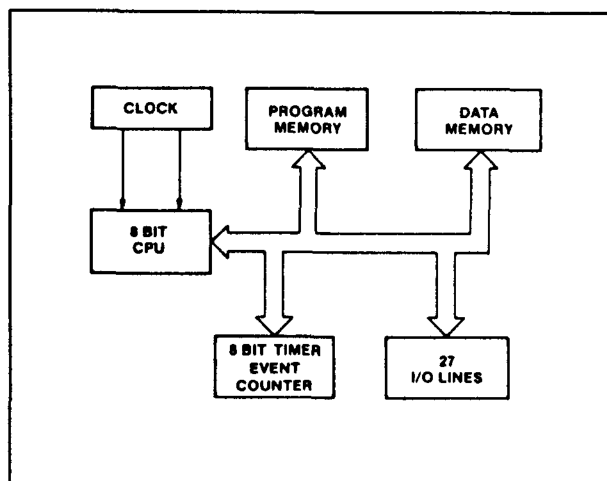
Kenmerken

- 8-bit parallelle computer
- Enkele +5 V +/-10 % voedingsspanning
- 40-pens DIL behuizing (figuur 7/6.1-31)
- Meer dan 96 instructies (waarvan 90 % één-byte)
- Alle instructies in 1 of 2 cyclussen uitvoerbaar
- Instructiecyclus minimaal 1,36 μs
- Interval timer/gebeurtenissen-teller
- Twee interrupts (één niveau)
- Compatibel met 8080/8085 periferieschakelingen
- Geheugen en I/O eenvoudig uit te breiden
- Geringe dissipatie
- 8039 en 8049 ook verkrijgbaar in CMOS: 80C39 en 80C49

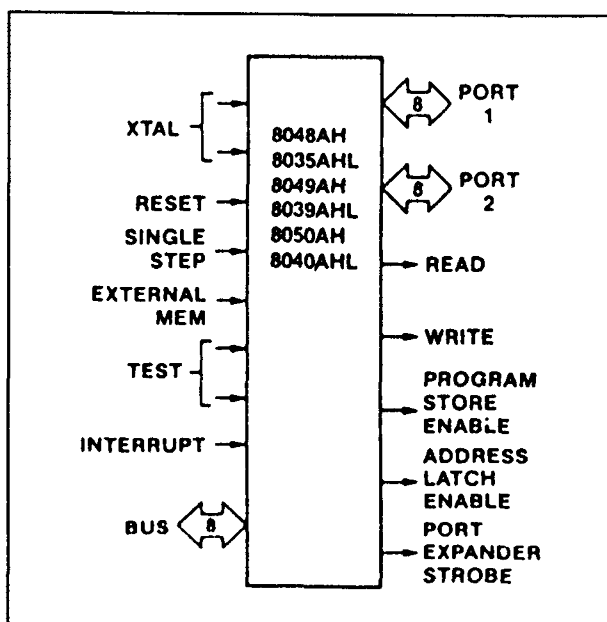
Overige kenmerken

De tabellen 7/6.1-14 tot en met 7/6.1-16 bevatten de overige elektrische en timing kenmerken van de 8048-familie, terwijl de bijbehorende golfvormen in de figuren 7/6.1-32 en -33 zijn opgenomen.

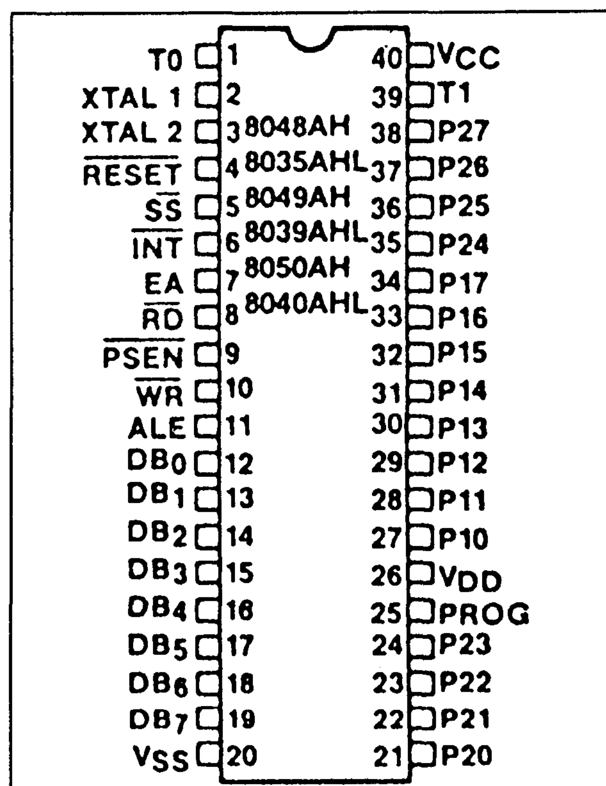
6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-29: Blokschema van de MCS-48 familie.



Figuur 7/6.1-30: Logisch symbool.



Figuur 7/6.1-31: Aansluitgegevens van de 8048-familie.

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to +150° C
Voltage On Any Pin With Respect to Ground	-0.5V to +7V
Power Dissipation	1.5 Watt

Tabel 7/6.1-14: Maximaal toegelaten waarden.

6.1 8-bit microcontrollers van de 8048-familie

Symbol	Parameter	Limits			Unit	Test Conditions	Device
		Min	Typ	Max			
V_{IL}	Input Low Voltage (All Except RESET, X1, X2)	-.5		.8	V		All
V_{IL1}	Input Low Voltage ($\overline{\text{RESET}}$, X1, X2)	-5		.6	V		All
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{\text{RESET}}$)	2.0		V_{CC}	V		All
V_{IH1}	Input High Voltage (X1, X2, $\overline{\text{RESET}}$)	3.8		V_{CC}	V		All
V_{OL}	Output Low Voltage (BUS)			.45	V	I_{OL} = 2.0 mA	All
V_{OL1}	Output Low Voltage ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{PSEN}}$, ALE)			.45	V	I_{OL} = 1.8 mA	All
V_{OL2}	Output Low Voltage (PROG)			.45	V	I_{OL} = 1.0 mA	All
V_{OL3}	Output Low Voltage (All Other Outputs)			.45	V	I_{OL} = 1.6 mA	All
V_{OH}	Output High Voltage (BUS)	2.4			V	I_{OH} = -400 μA	All
V_{OH1}	Output High Voltage ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{PSEN}}$, ALE)	2.4			V	I_{OH} = -100 μA	All
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = -40 μA	All
I_{L1}	Leakage Current (T1, $\overline{\text{INT}}$)			± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$	All
I_{LI1}	Input Leakage Current (P10-P17, P20-P27, EA, $\overline{\text{SS}}$)			−500	μA	$V_{SS} + .45 \leq V_{IN} \leq V_{CC}$	All
I_{LI2}	Input Leakage Current $\overline{\text{RESET}}$	20		300	μA	$V_{SS} \leq V_{IN} \leq 3.8V$	All
I_{LO}	Leakage Current (BUS, T0) (High Impedance State			± 10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$	All
I_{DD}	V_{DD} Supply Current (RAM Standby)		3	5	mA		8048AH 8035AHL
			4	7	mA		8049AH 8039AHL
			5	10	mA		8050AH 8040AHL
I_{DD} + I_{CC}	Total Supply Current*		30	65	mA		8048AH 8035AHL
			35	70	mA		8049AH 8039AHL
			40	80	mA		8050AH 8040AHL
V_{DD}	RAM Standby Voltage	2.2		5.5	V	Standby Mode Reset $\leq V_{IL1}$	All

*$I_{CC} + I_{DD}$ is measured with all outputs disconnected; $\overline{\text{SS}}$, $\overline{\text{RESET}}$, and $\overline{\text{INT}}$ equal to V_{CC}; EA equal to V_{SS}.

Tabel 7/6.1-15: Gelijkspanningscondities voor de zes typen.

6.1 8-bit microcontrollers van de 8048-familie

A.C. CHARACTERISTICS: (T_A = 0°C to 70°C; V_{CC} = V_{DD} = 5V ± 10%; V_{SS} = 0V)

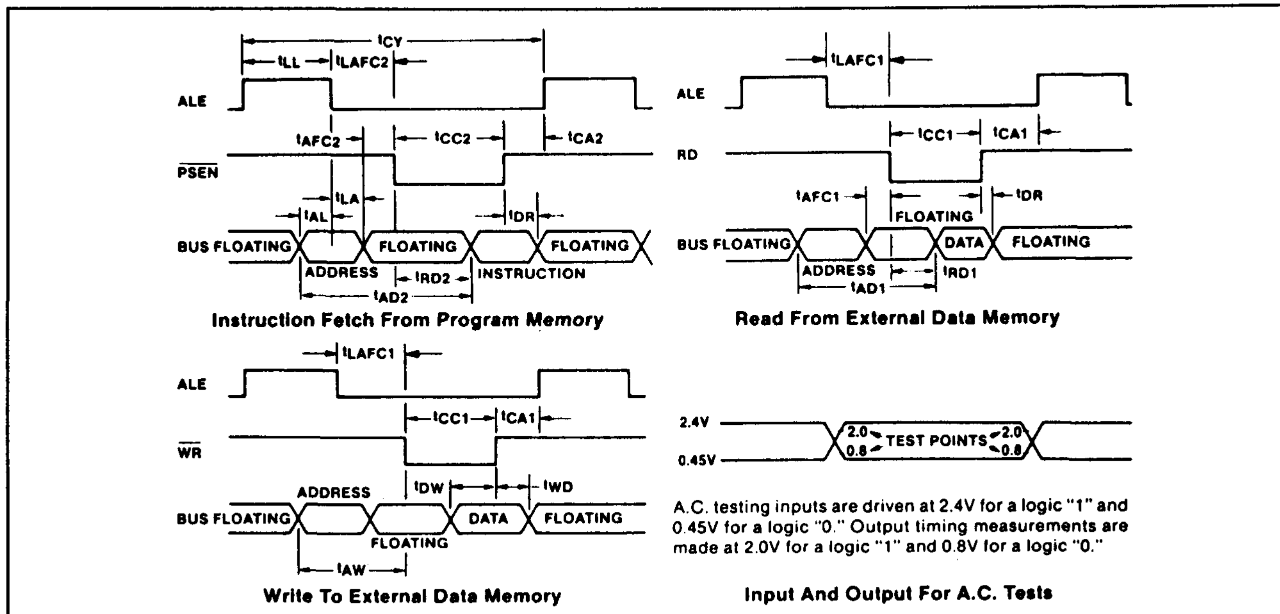
Symbol	Parameter	f (t) (Note 3)	11 MHz		Unit	Conditions (Note 1)
			Min	Max		
t	Clock Period	1/xtal freq	90.9	1000	ns	(Note 3)
t_{LL}	ALE Pulse Width	3.5t-170	150		ns	
t_{AL}	Addr Setup to ALE	2t-110	70		ns	(Note 2)
t_{LA}	Addr Hold from ALE	t-40	50		ns	
t_{CC1}	Control Pulse Width ($\overline{RD}$, $\overline{WR}$)	7.5t-200	480		ns	
t_{CC2}	Control Pulse Width ($\overline{PSEN}$)	6t-200	350		ns	
t_{DW}	Data Setup before $\overline{WR}$	6.5t-200	390		ns	
t_{WD}	Data Hold after $\overline{WR}$	t-50	40		ns	
t_{DR}	Data Hold ($\overline{RD}$, $\overline{PSEN}$)	1.5t-30	0	110	ns	
t_{RD1}	$\overline{RD}$ to Data in	6t-170		375	ns	
t_{RD2}	$\overline{PSEN}$ to Data in	4.5t-170		240	ns	
t_{AW}	Addr Setup to $\overline{WR}$	5t-150	300		ns	
t_{AD1}	Addr Setup to Data ($\overline{RD}$)	10.5t-220		730	ns	
t_{AD2}	Addr Setup to Data ($\overline{PSEN}$)	7.5t-200		460	ns	
t_{AFC1}	Addr Float to $\overline{RD}$, $\overline{WR}$	2t-40	140		ns	(Note 2)
t_{AFC2}	Addr Float to $\overline{PSEN}$	.5t-40	10		ns	(Note 2)
t_{LAFC1}	ALE to Control ($\overline{RD}$, $\overline{WR}$)	3t-75	200		ns	
t_{LAFC2}	ALE to Control ($\overline{PSEN}$)	1.5t-75	60		ns	
t_{CA1}	Control to ALE ($\overline{RD}$, $\overline{WR}$, $\overline{PROG}$)	t−65	25		ns	
t_{CA2}	Control to ALE ($\overline{PSEN}$)	4t−70	290		ns	
t_{CP}	Port Control Setup to $\overline{PROG}$	1.5t-80	50		ns	
t_{PC}	Port Control Hold to $\overline{PROG}$	4t-260	100		ns	
t_{PR}	$\overline{PROG}$ to P2 Input Valid	8.5t-120		650	ns	
t_{PF}	Input Data Hold from $\overline{PROG}$	1.5t	0	140	ns	
t_{DP}	Output Data Setup	6t-290	250		ns	
t_{PD}	Output Data Hold	1.5t-90	40		ns	
t_{PP}	$\overline{PROG}$ Pulse Width	10.5t-250	700		ns	
t_{PL}	Port 2 I/O Setup to ALE	4t-200	160		ns	
t_{LP}	Port 2 I/O Hold to ALE	.5t-30	15		ns	
t_{PV}	Port Output from ALE	4.5t+100		510	ns	
t_{OPRR}	T0 Rep Rate	3t	270		ns	
t_{CY}	Cycle Time	15t	1.36	15.0	µs	

Notes:

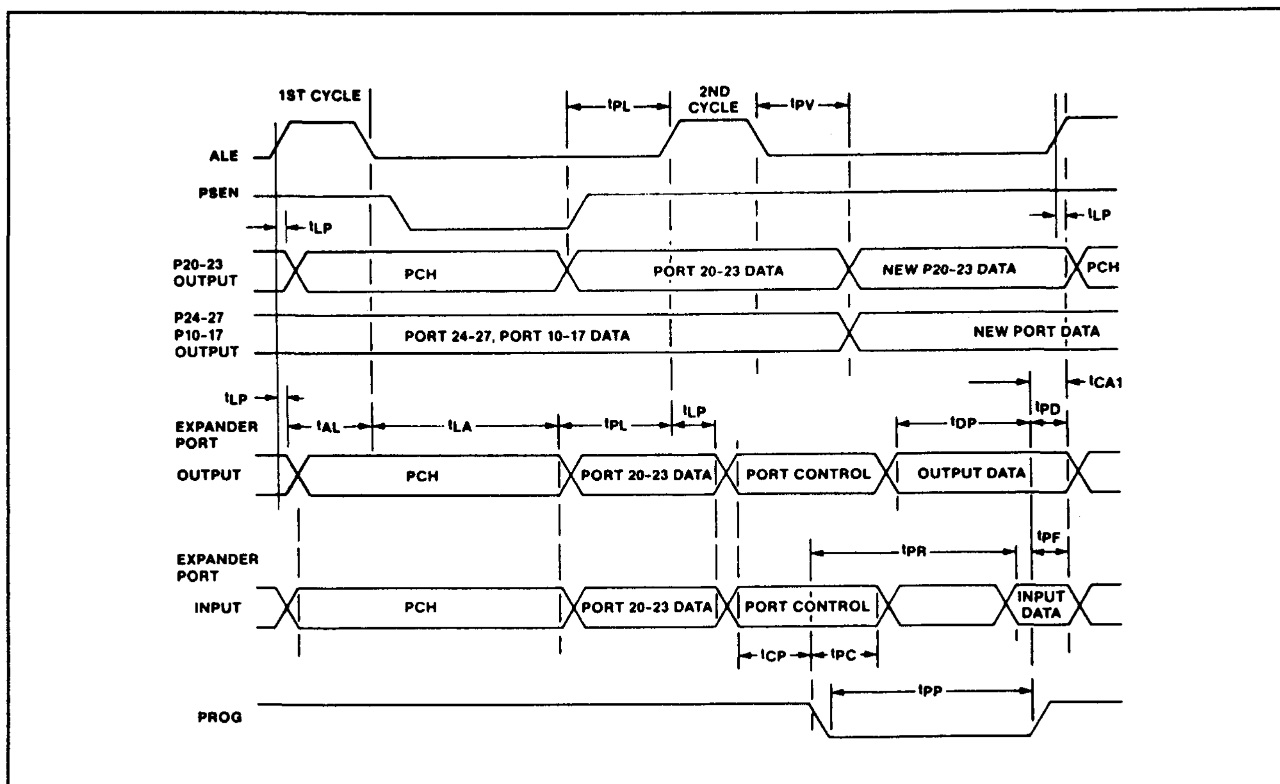
1. Control Outputs CL = 80pF
   BUS Outputs CL = 150pF
2. BUS High Impedance
   Load 20pF.
3. f(t) assumes 50% duty cycle on X1, X2. Max clock period is for a 1 MHz crystal input.

Tabel 7/6.1-16: Periode- en schakeltijden voor alle zes typen.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-32: Golfvormen (zie ook tabel 7/6.1-16).

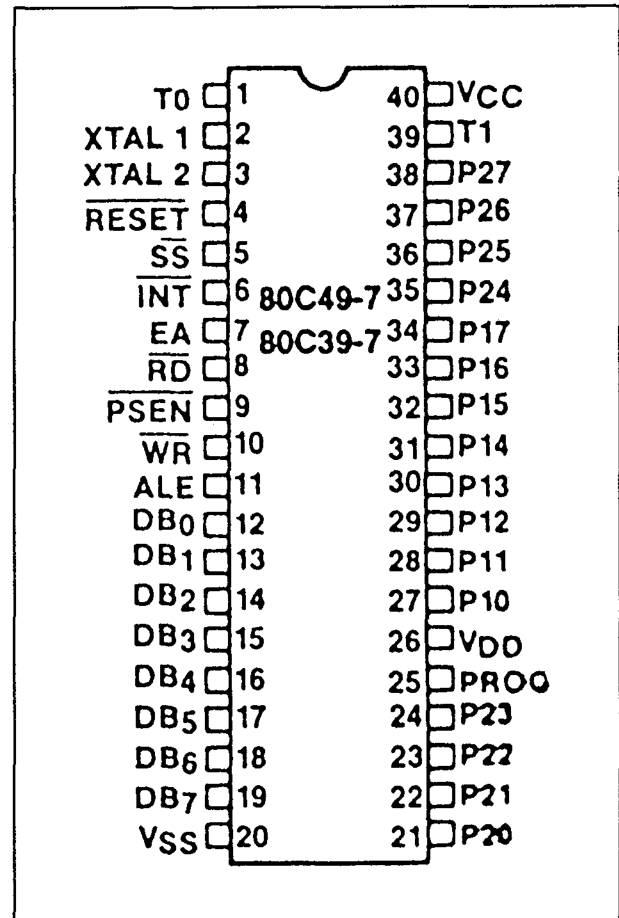


Figuur 7/6.1-33: Timingen ten behoeve van de poorten 1 en 2.

80C49 (met ROM)
80C39 (zonder ROM)
CMOS microcontroller (single-chip 8-bit microprocessor)

Van de 8039 (ROM-loos) en 8049 (met ROM) zijn van diverse fabrikanten ook CMOS-versies verkrijgbaar die minder vermogen dissiperen en geschikt zijn voor een uitgebreider voedingsspanningengebied. Bij power-down nemen deze typen slechts 2 µA uit de voeding (2 V) op. De aansluitingen en werking zijn identiek aan die van de NMOS-typen, zodat daarnaar en naar het algemene gedeelte verwezen wordt (tabel 7/6.1-16 en de figuren 7/6.1-29 tot en met 7/6.1-33).
In dit gedeelte worden uitsluitend de tabellen met de specifieke eigenschappen van de CMOS-uitvoeringen opgenomen (tabel 7/6.1-17 en -18) en voor het gemak de aansluitgegevens, figuur 7/6.1-34.

Kenmerken
- Pen-compatibel met de 8039 en 8049
- Geschikt voor batterijvoeding
- 3 Dissipatie-mogelijkheden:
  Normaal: 12 mA/5 V (11 MHz)
  Vrijloop: 5 mA/5 V (11 MHz)
  Power Down: 2 µA/2 V



Figuur 7/6.1-34: Aansluitingen van de 80C39 en de 80C49.

8748, 8749 (met EPROM)
NMOS microcontroller (single-chip 8-bit microprocessor met EPROM)

De 8748 en 8749 zijn met EPROM uitgeruste versies van de 8048 en 8049 (met respectievelijk 1 kB x 8 en 2 kB x 8 EPROM aan boord). Deze typen hebben, op de programmeerbare en wisbare EPROM na, dezelfde eigenschappen als de andere controllers uit de MCS-48 familie. Zij zijn onder andere zeer geschikt voor prototypen en kleine series.
In tabel 7/6.1-19 worden de 8748 en 8749 vergeleken met de ROM-loze versies 8035 en 8039, terwijl in figuur 7/6.1-35 de aansluitgegevens worden getoond. De timing en schakeltijden zijn identiek aan die van de 8048/8049 (zie tabel 7/6.1-16 en bijbehorende figuren 7/6.1-32 en -33).

ABSOLUTE MAXIMUM RATINGS

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	−65°C to +150°C
Voltage On Any Pin With Respect to Ground	−0.5V to V_{CC} + 1V
Maximum Voltage On Any Pin With Respect to Ground	7V
Power Dissipation	1.0 Watt

Tabel 7/6.1-17: Maximaal toegelaten waarden.

6.1 8-bit microcontrollers van de 8048-familie

D.C. CHARACTERISTICS: (T_A = 0°C to 70°C; $V_{CC} = V_{DD}$ = 5V ± 20%; $|V_{CC} - V_{DD}| \leq$ 1.5V; V_{SS} = 0V)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
V_{IL}	Input Low Voltage (All Except X1, $\overline{RESET}$)	−.5		.18 V_{CC}	V	
V_{IL1}	Input Low Voltage X1, $\overline{RESET}$	−5		.13 V_{CC}	V	
V_{IH}	Input High Voltage (All Except XTAL1, $\overline{RESET}$)	0.2 V_{CC} + 1.2		V_{CC}	V	
V_{IH1}	Input High Voltage (X1, $\overline{RESET}$)	.7 V_{CC}		V_{CC}	V	
V_{OL}	Output Low Voltage (BUS)			.6	V	I_{OL} = 2.0 mA
V_{OL1}	Output Low Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)			.6	V	I_{OL} = 1.8 mA
V_{OL2}	Output Low Voltage (PROG)			.6	V	I_{OL} = 1.0 mA
V_{OL3}	Output Low Voltage (All Other Outputs)			.6	V	I_{OL} = 1.6 mA
V_{OH}	Output High Voltage (BUS)	.75 V_{CC}			V	I_{OH} = −400 μA
V_{OH1}	Output High Voltage ($\overline{RD}$, $\overline{WR}$, $\overline{PSEN}$, ALE)	.75 V_{CC}			V	I_{OH} = −100 μA
V_{OH2}	Output High Voltage (All Other Outputs)	2.4 3.0			V	I_{OH} = −40 μA I_{OH} = −20 μA
I_{L1}	Input Leakage Current (T1, $\overline{INT}$, EA)			±5	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LI1}	Input Leakage Current (P10–P17, P20–P27, $\overline{SS}$)			−500	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LO}	Output Leakage Current (BUS, TO) (High Impedance State)			±5	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{LR}	Input Leakage Current ($\overline{RESET}$)	−10		−300	μA	$V_{SS} \leq V_{IN} \leq V_{IH1}$
I_{PD}	Power Down Standby Current			2	μA	V_{DD} = 2.0V $\overline{RESET} \leq V_{IL}$

I_{CC} Active Current (mA)

V_{CC}	4V	5V	6V
1 MHz	2.5	3.3	4.0
6 MHz	5	6.8	8.5
11 MHz	9	12	15

I_{CC} Idle Current (mA)

V_{CC}	4V	5V	6V
1 MHz	1.7	2.0	2.2
6 MHz	2	3	4
11 MHz	3.5	4.8	6

Absolute Maximum Unloaded Current

I_{CC} Test Conditions:

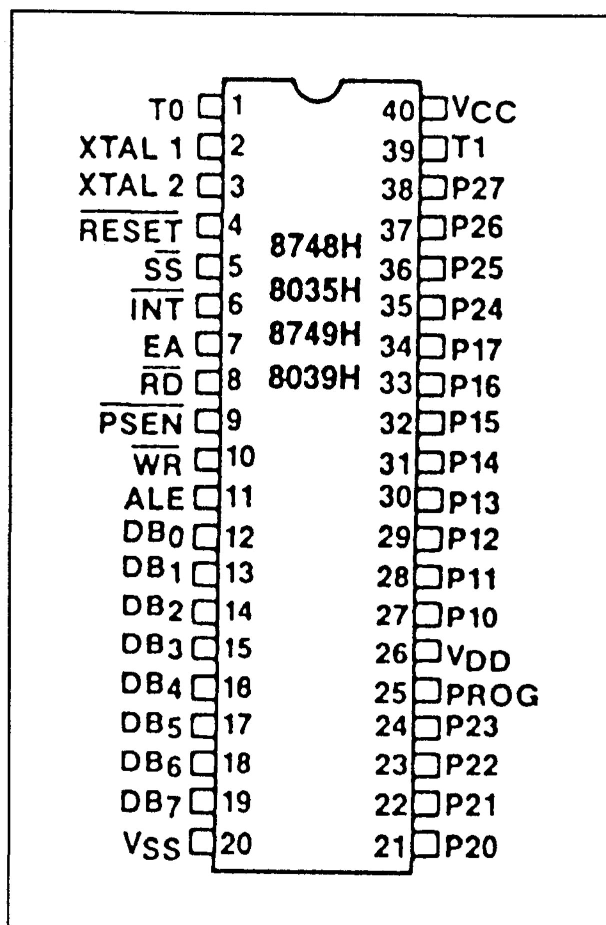
I_{CC} Active
All outputs disconnected
T1, $\overline{INT}$, SS, T0 connected to HIGH (V_{IH})
EA, $\overline{RST}$ connected to LOW (V_{IL})
XTAL1 External Drive
Rise Time = 10 ns, Fall Time = 10 ns
XTAL2 No connection
$V_{IH} = V_{CC}$ − 0.5V
$V_{IL} = V_{SS}$ + 0.5V

I_{CC} Idle
All outputs disconnected
XTAL1 External Drive
Rise Time = 10 ns, Fall Time = 10 ns
XTAL2 No connection
$V_{IH} = V_{CC}$ − 0.5V
$V_{IL} = V_{SS}$ + 0.5V

Tabel 7/6.1-18: Gelijkspanningscondities.

In dit gedeelte worden alleen de afwijkende kenmerken, met name de gelijkspanningscondities (tabel 7/6.1-20) en het programmeren/verifiëren van de EPROM (tabellen 7/6.1-21 en -22 en de figuren 7/6.1-36 en -37) vermeld.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-35: Aansluitingen van de EPROM-versies 8748 en 8749.

Programmeren, verifiëren en wissen van de EPROM

De algemene gang van zaken bij wissen, programmeren en verifiëren van de EPROM's in de 8748 en 8749 is reeds aan het begin van dit deel, in het algemene gedeelte behandeld.

Hier worden de exacte spanningen, stromen en tijden vermeld.

In het kort komt het programmeerproces neer op het aktief maken van de programmeermode, een adres toevoeren, dit adres in de latch opslaan, data toevoeren en een programmeerpuls geven. Elk woord wordt compleet geprogrammeerd en geverifiëerd voordat naar het volgende woord wordt overgegaan.

In tabel 7/6.1-21 zijn de pennen vermeld die voor het programmeren worden gebruikt.

De tabellen 7/6.1-22 en -23 tonen tenslotte de elektrische specificaties en de timing die bij het programmeren in acht genomen dienen te worden.

Figuur 7/6.1-36 laat de in tabel 7/6.1-23 vermelde tijden en de golfvormen bij programmeren en verifiëren van de EPROM's zien, terwijl figuur 7/6.1-37 alleen de verificatie toont.

Device	Internal Memory	
8039H	none	128 x 8 RAM
8035H	none	64 x 8 RAM
8749H	2K x 8 EPROM	128 x 8 RAM
8748H	1K x 8 EPROM	64 x 8 RAM

Tabel 7/6.1-19: Vergelijking van de EPROM-versies 8748 en 8749 met de ROM-loze typen 8035 en 8039.

6.1 8-bit microcontrollers van de 8048-familie

D.C. CHARACTERISTICS: (T_A = 0°C to 70°C; V_{CC} = V_{DD} = 5V ± 10%; V_{SS} = 0V)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	Device
V_{IL}	Input Low Voltage (All Except RESET, X1, X2)	-.5		.8	V		All
V_{IL1}	Input Low Voltage (RESET, X1, X2)	-.5		.6	V		All
V_{IH}	Input High Voltage (All Except XTAL1, XTAL2, RESET)	2.0		V_{CC}	V		All
V_{IH1}	Input High Voltage (X1, X2, RESET)	3.8		V_{CC}	V		All
V_{OL}	Output Low Voltage (BUS)			.45	V	I_{OL} = 2.0 mA	All
V_{OL1}	Output Low Voltage (RD, WR, PSEN, ALE)			.45	V	I_{OL} = 1.8 mA	All
V_{OL2}	Output Low Voltage (PROG)			.45	V	I_{OL} = 1.0 mA	All
V_{OL3}	Output Low Voltage (All Other Outputs)			.45	V	I_{OL} = 1.6 mA	All
V_{OH}	Output High Voltage (BUS)	2.4			V	I_{OH} = -400 μA	All
V_{OH1}	Output High Voltage (RD, WR, PSEN, ALE)	2.4			V	I_{OH} = -100 μA	All
V_{OH2}	Output High Voltage (All Other Outputs)	2.4			V	I_{OH} = -40 μA	All
I_{L1}	Leakage Current (T1, INT)			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$	All
I_{LI1}	Input Leakage Current (P10–P17, P20–P27, EA, SS)			−500	μA	$V_{SS}+.45 \leq V_{IN} \leq V_{CC}$	All
I_{LI2}	Input Leakage Current RESET	−10		−300	μA	$V_{SS} \leq V_{IN} \leq 3.8V$	All
I_{LO}	Leakage Current (BUS, T0) (High Impedance State)			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$	All
I_{DD} + I_{CC}	Total Supply Current*		80	100	mA		8035H
			95	110	mA		8039H
			80	100	mA		8748H
			95	110	mA		8749H

*I_{CC} + I_{DD} is measured with all outputs disconnected; SS, RESET, and INT equal to V_{CC}; EA equal to V_{SS}.

Tabel 7/6.1-20: Gelijkspanningscondities.

6.1 8-bit microcontrollers van de 8048-familie

Pin	Function
XTAL 1 XTAL 2	Clock Input (3 to 4.0 MHz)
Reset	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-P22	Address Input
V_{DD}	Programming Power Supply
PROG	Program Pulse Input

Tabel 7/6.1-21: De bij het programmeren gebruikte pennen en hun functies.

De programmeer/verifieer-volgorde is:
- V_{DD} = 5 V. De inwendige oscillator werkt of een extern clocksignaal wordt toegevoerd.
  RESET = 0 V, TEST 0 = 5 V, EA = 5 V, BUS en PROG zijn zwevend. P10 en P11 liggen aan aarde.
- Doe de 8748 (8749) in de programmeervoet.
- TEST 0 = 0 V (kies programmeermode).
- EA = 18 V (aktiveer programmeermode).
- Zet adres op BUS en P20 tot en met P22.
- RESET = 5 V (latch adres).
- Zet data op BUS.
- V_{DD} = 21 V (programmeerspanning).
- PROG = V_{CC} of zwevend, gevolgd door een 50 ms durende puls met een amplitude van 18 V.
- V_{DD} = 5 V.
- TEST 0 = 5 V (verificatiemode).
- Lees en verifieer data op BUS.
- TEST 0 = 0 V.
- RESET = 0 V en herhaal vanaf stap 5.
- Wanneer de 8748 (8749) uit de voet wordt verwijderd moeten de condities van stap 1 worden aangebracht.

Waarschuwing

Wanneer wordt geprobeerd een foutief geplaatste 8748 (8749) te programmeren, kan dit onderdeel zwaar worden beschadigd. De aanwezigheid van het ALE clock-signaal is een indicatie dat het onderdeel zich correct in het voetje bevindt.

D.C. TIMING SPECIFICATION FOR PROGRAMMING 8748H/8749H ONLY:
(T_A = 25°C ± 5°C; V_{CC} = 5V ± 5%; V_{DD} = 21 ± .5V)

Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{DDH}	V_{DD} Program Voltage High Level	20.5	21.5	V	
V_{DDL}	V_{DD} Voltage Low Level	4.75	5.25	V	
V_{PH}	PROG Program Voltage High Level	17.5	18.5	V	
V_{PL}	PROG Voltage Low Level	4.0	V_{CC}	V	
V_{EAH}	EA Program or Verify Voltage High Level	17.5	18.5	V	
I_{DD}	V_{DD} High Voltage Supply Current		20.0	mA	
I_{PROG}	PROG High Voltage Supply Current		1.0	mA	
I_{EA}	EA High Voltage Supply Current		1.0	mA	

Tabel 7/6.1-22: Gelijkspanningscondities bij het programmeren.

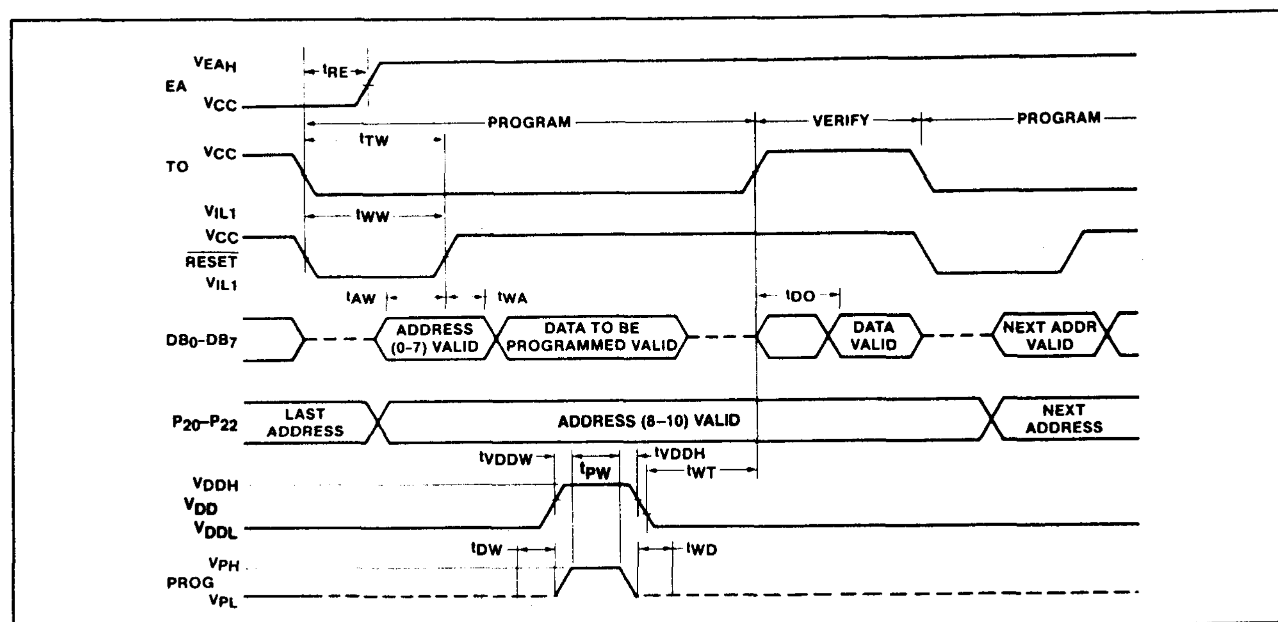
6.1 8-bit microcontrollers van de 8048-familie

A.C. TIMING SPECIFICATION FOR PROGRAMMING 8748H/8749H ONLY:
(T_A = 25° C ± 5° C; V_{CC} = 5V ± 5%; V_{DD} = 21 ± .5V)

Symbol	Parameter	Min	Max	Unit	Test Conditions
t_{AW}	Address Setup Time to $\overline{\text{RESET}}$↑	$4t_{CY}$			
t_{WA}	Address Hold Time After $\overline{\text{RESET}}$↑	$4t_{CY}$			
t_{DW}	Data in Setup Time to PROG↑	$4t_{CY}$			
t_{WD}	Data in Hold Time After PROG↓	$4t_{CY}$			
t_{PH}	$\overline{\text{RESET}}$ Hold Time to Verify	$4t_{CY}$			
t_{VDDW}	V_{DD} Hold Time Before PROG↑	0	1.0	ms	
t_{VDDH}	V_{DD} Hold Time After PROG↓	0	1.0	ms	
t_{PW}	Program Pulse Width	50	60	ms	
t_{TW}	Test 0 Setup Time for Program Mode	$4t_{CY}$			
t_{WT}	Test 0 Hold Time After Program Mode	$4t_{CY}$			
t_{DO}	Test 0 to Data Out Delay		$4t_{CY}$		
t_{WW}	$\overline{\text{RESET}}$ Pulse Width to Latch Address	$4t_{CY}$			
t_r, t_f	V_{DD} and PROG Rise and Fall Times	0.5	100	μs	
t_{CY}	CPU Operation Cycle Time	3.75	5	μs	
t_{RE}	$\overline{\text{RESET}}$ Setup Time before EA↑	$4t_{CY}$			

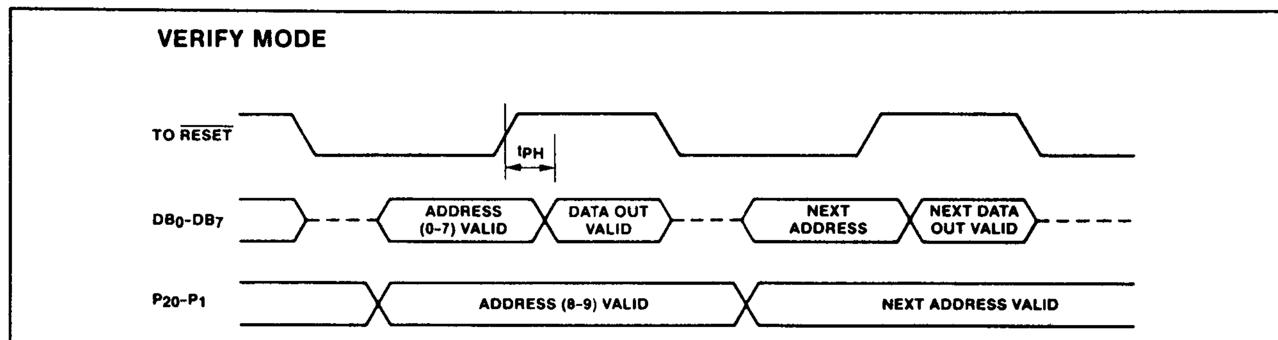
NOTE: If Test 0 is high, t_{DO} can be triggered by $\overline{\text{RESET}}$↑.

Tabel 7/6.1-23: Timing bij het programmeren van de 8748 (8749).



Figuur 7/6.1-36: Golfvormen en schakeltijden bij programmeren en verifiëren van de EPROM's.

6.1 8-bit microcontrollers van de 8048-familie



Figuur 7/6.1-37: Golfvormen en schakeltijden bij verificatie.

6.1 8-bit microcontrollers van de 8048-familie

7/6.2

Algemene eigenschappen van de 8051-familie

Inleiding

Familie-overzicht

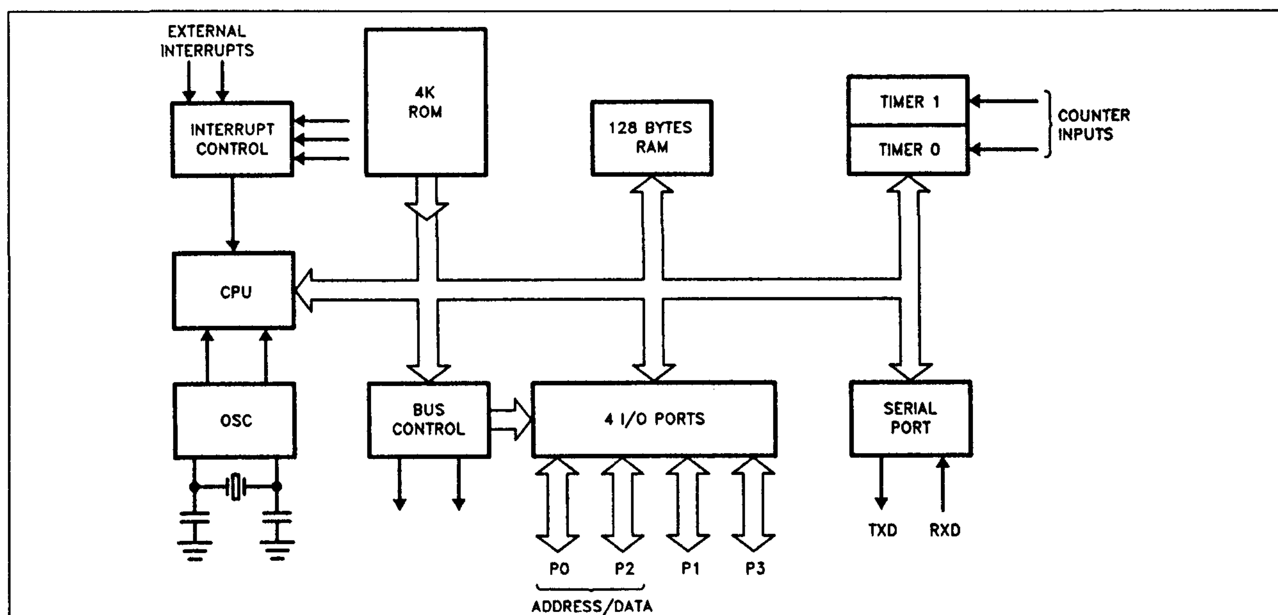
Rond de kern van de 8051 microcontroller van Intel is een complete familie (de MCS-51) opgebouwd, die ondertussen al uit vele tientallen leden bestaat. Deze veel gebruikte controllers worden door verschillende fabrikanten (Intel, Philips, Siemens, Oki, Matra Harris) geleverd, waarbij ze al naar gelang de toepassing zijn voorzien van allerlei interfaces, zowel digitaal als analoog, en aangepast inwendig geheugen. De eerste NMOS-typen zijn opgevolgd door snellere en zuiniger CMOS-typen, terwijl nu ook EEPROM als intern geheugen ter beschikking staat. Bovendien zijn nu 3 V typen leverbaar.

Om de beschrijving niet al te ingewikkeld te maken worden in dit hoofdstuk de architectuur en de instructieset van de oorspronkelijke leden van de familie behandeld. De beschrijving van de afzonderlijke typen, ook de veel gebruikte specifieke, inclusief uitbreidingen en afwijkingen volgt daarna.

Eigenschappen

De 8051-kern heeft de volgende eigenschappen (zie ook figuur 7/6.2-1):

- 8-bit CPU, geoptimaliseerd voor besturingstoepassingen



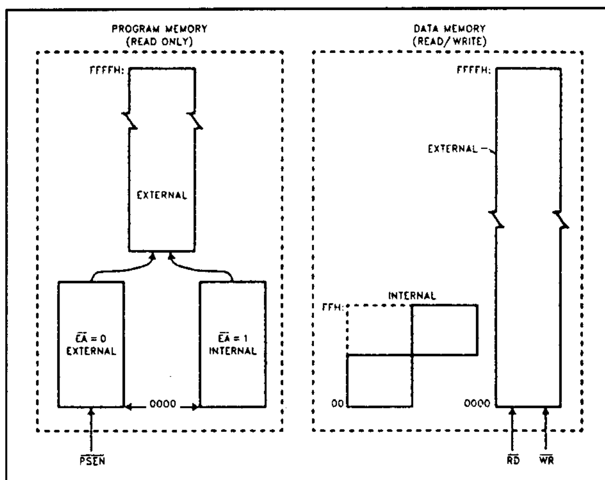
Figuur 7/6.2-1: Vereenvoudigd blokschema van de 8051-kern.

6.2 Algemene eigenschappen van de 8051-familie

Device Name	ROMless Version	EPROM Version	ROM Bytes	RAM Bytes	16-bit Timers	Ckt Type
8051AH	8031AH	8751H, 8751BH	4K	128	2	HMOS
8052AH	8032AH	8752BH	8K	256	3	HMOS
80C51BH	80C31BH	87C51	4K	128	2	CHMOS

Tabel 7/6.2-1: De basistypen en belangrijkste kenmerken van de 8051-serie.

- uitgebreide Boole'se mogelijkheden (enkele bit logika)
- 64 kB adresruimte voor programmageheugen
- 64 kB adresruimte voor data-geheugen
- 4 kB programmageheugen op de chip
- 128 bytes data-RAM op de chip
- 32 bidirectionele en individueel adresseerbare I/O-lijnen (vier 8 bit poorten)
- twee 16 bit timer/counters
- full duplex UART
- 6 source 5 vector interruptstructuur met twee prioriteitsniveaus
- clock oscillator op de chip



Figuur 7/6.2-2: Geheugenstructuur van de 8051-familie.

In tabel 7/6.2-1 is een overzicht te zien van de oorspronkelijke 8051-familie. De instructieset en de aansluitingen zijn voor alle leden gelijk. Intern verschillen ze van elkaar door de grootte van het RAM-geheugen, het al of niet aanwezig zijn van EPROM of ROM en het aantal tellers. De 8031, 8051 en 8751 (ook de CMOS-uitvoeringen) bevatten 128 bytes RAM en respectievelijk geen, 4 k x 8 ROM en 4 k x 8 EPROM. De 8032, 8052 en 8752 (ook CMOS) hebben tweemaal zoveel RAM en (E)PROM.

De typen zonder (EP)ROM kunnen worden gebruikt voor het ontwikkelen van systemen, terwijl de typen met EPROM zich het beste lenen voor kleine series "embedded" systemen en prototypen. De typen met maskerprogrammeerbaar ROM zijn uiteraard bedoeld voor grote series identieke schakelingen.

Geheugen organisatie

Inleiding

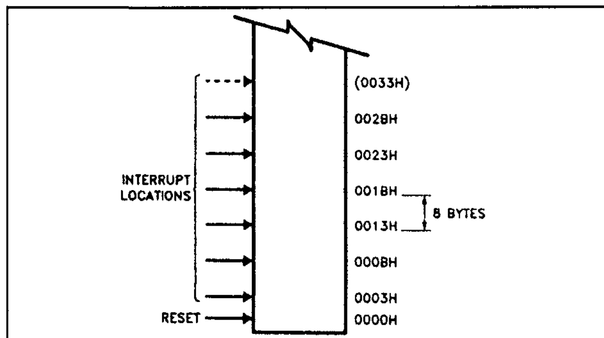
Alle leden van de MCS-51 (8051) familie hebben aparte adresruimten voor programma en data (zie ook figuur 7/6.2-2).

Door deze logische scheiding kan toegang worden verkregen tot het datageheugen met 8 bit adressen, zodat de 8 bit processor deze sneller kan bewerken. Door het DPTR register kunnen echter ook 16 bit adressen worden gegenereerd.

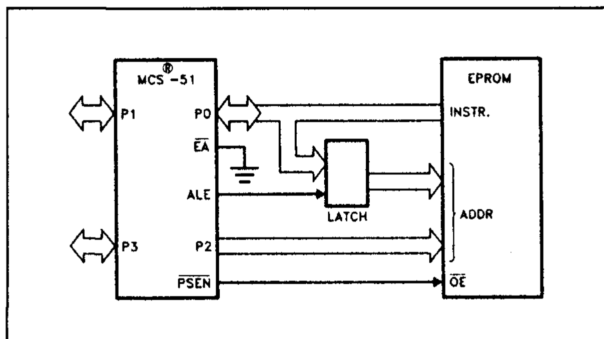
Het programmageheugen is maximaal 64 kB groot en kan alleen worden uitgelezen. In de ROM en EPROM versies bevinden de laagste 4 , 8 of 16 kB zich op de chip; in de ROM-loze versie is het gehele programmageheugen extern. De lees-strobe voor extern programmageheugen is het signaal $\overline{PSEN}$ (Program Store Enable).

6.2 Algemene eigenschappen van de 8051-familie

Het datageheugen bezet een aparte adresruimte. Maximaal 64 kB extern RAM-geheugen kan worden geadresseerd. De CPU genereert de lees- en schrijfsignalen $\overline{RD}$ en $\overline{WR}$.
Eventueel kunnen extern programma- en datageheugen worden gecombineerd door de $\overline{RD}$ en $\overline{PSEN}$ signalen op een AND-poort aan te sluiten en de uitgang van deze poort te gebruiken als de leesstrobe naar het externe programma/datageheugen.



Figuur 7/6.2-3: Programmageheugen van de 8051-familie.



Figuur 7/6.2-4: Het uitlezen van extern programmageheugen.

Programmageheugen

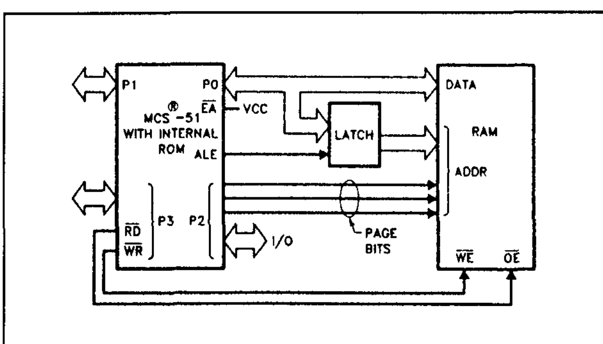
Figuur 7/6.2-3 toont het onderste deel van het programmageheugen. Na het resetten begint de CPU op lokatie 0000H (0 hexadecimaal). Zoals in deze figuur ook te zien is, is elke interrupt toegewezen aan een vaste lokatie in het programmageheugen. De interrupt laat de CPU naar zo'n lokatie springen, waar vandaan de serviceroutine wordt uitgevoerd. De externe interrupt 0 is bijvoorbeeld toegewezen aan lokatie 0003H. Indien een interrupt niet gebruikt gaat worden, is zijn servicelokatie beschikbaar als algemeen programmageheugen. De interrupt servicelokaties liggen telkens 8 byte uit elkaar: 0003H voor Externe Interrupt 0, 000BH voor Timer 0, 0013H voor Externe Interrupt 1, 001BH voor Timer 1, enzovoorts. Als een interrupt serviceroutine kort genoeg is (hetgeen bij besturingen vaak het geval is), kan hij in zijn geheel binnen het 8 byte interval liggen. Bij langere serviceroutines wordt een jumpinstructie gebruikt om over de andere interruptlokaties te springen. De laagste 4 of 8 of 16 kB van het programmageheugen kunnen zich zowel in het interne ROM als in een extern ROM bevinden. Deze keuze wordt gemaakt door de $\overline{EA}$-pen (External Access) aan V_{CC} of aan V_{SS} te leggen. In de 4 kB ROM-typen worden, als de $\overline{EA}$-pen aan V_{CC} ligt, uitlezingen op het interne geheugen uitgevoerd vanaf adres 0000H tot en met 0FFFH. Uitlezingen van adres 1000H tot en met FFFFH worden gericht op extern geheugen. In de 8 kB ROM-typen worden adressen tussen 0000H en 1FFFH als intern beschouwd (bij $\overline{EA} = V_{CC}$) en adressen vanaf 2000H tot en met FFFFH als extern. Bij de 16 kB ROM-typen loopt de interne adresruimte van 0000H tot en met 3FFFH.
Als de $\overline{EA}$-pen aan V_{SS} ligt, worden alle programma uitlezingen gedaan op extern ROM-geheugen. De ROM-loze typen werken alleen correct als deze pen aan V_{SS} ligt. De lees-strobe $\overline{PSEN}$ voor extern ROM wordt voor alle externe programma uitlezingen gebruikt. Voor interne uitlezingen wordt $\overline{PSEN}$ niet geaktiveerd. De hardware configuratie voor het uitlezen van extern programma is te zien in figuur 7/6.2-4. Merk op dat hierbij 16 in-/uitgangslijnen (poorten 0 en 2) in gebruik zijn voor busfuncties. Poort 0 (P0) dient als een gemultiplexte adres/databus. Hij zendt de lage byte van de Program Counter (PCL) als adres uit, waarna hij in een zwevende toestand gaat in afwachting van

6.2 Algemene eigenschappen van de 8051-familie

de codebyte uit het programmageheugen. Gedurende de tijd dat het lage byte van de Program Counter geldig is op P0, wordt dit byte door het ALE-signaal (Address Latch Enable) in een adreslatch geklokt. Ondertussen zendt poort 2 (P2) het hoge byte van de Program Counter (PCH) uit. Daarna ontvangt de EPROM het $\overline{PSEN}$-signaal en wordt de code door de microcontroller ingelezen. De adressen van het programmageheugen zijn altijd 16 bit breed, ook als minder dan 64 kB wordt gebruikt. Voor het gebruik van externe programma's zijn dus twee van de 8 bit poorten nodig.

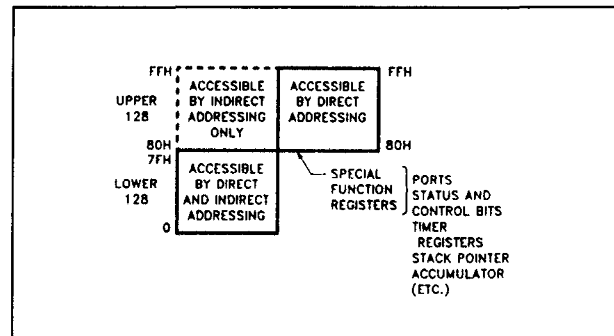
Datageheugen

Aan de rechter kant van figuur 7/6.2-2 zijn de interne en externe datageheugenruimten te zien die de gebruiker ter beschikking staan. In figuur 7/6.2-5 is de hardware configuratie te zien die nodig is om maximaal 2 kB extern RAM te bereiken. De CPU werkt in dit geval met intern ROM. Poort 0 dient als een gemultiplexte adres/databus voor de RAM, terwijl drie lijnen van poort 2 worden gebruikt om de RAM te pagineren. De CPU genereert de benodigde $\overline{RD}$ en $\overline{WR}$ signalen.



Figuur 7/6.2-5: Het bereiken van extern datageheugen.

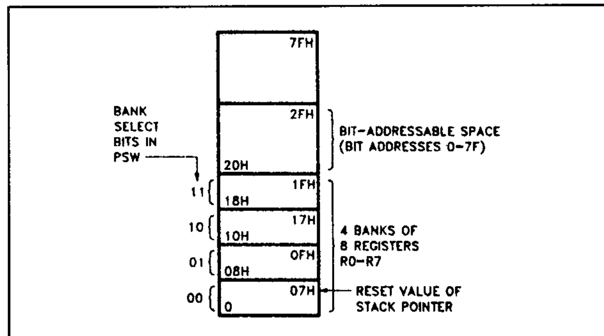
Er kan maximaal 64 kB extern datageheugen worden gebruikt, waarvan de adressen 1 of 2 byte breed kunnen zijn. Eén byte adressen worden vaak samen met andere I/O-lijnen gebruikt om de RAM te pagineren (zie ook figuur 7/6.2-5). Wanneer twee byte adressen worden gebruikt, wordt het hoge adresbyte op poort 2 gezet.



Figuur 7/6.2-6: Intern datageheugen.

Figuur 7/6.2-6 toont de organisatie van het interne datageheugen. De geheugenruimte is verdeeld in drie blokken, die meestal het "Lower-128", het "Upper-128" en "SFR-ruimte" worden genoemd. De adressen van het interne datageheugen zijn altijd één byte breed, waardoor de adresruimte slechts 256 bytes kan zijn. Door een eenvoudige truc zijn de adresseringsmodes voor intern RAM echter geschikt voor 384 bytes: directe adressen hoger dan 7FH beslaan één geheugenruimte en indirecte adressen hoger dan 7FH een andere geheugenruimte. Daarom beslaan in figuur 7/6.2-6 het Upper-128 en de SFR-ruimte hetzelfde adresblok 80H tot en met FFH, terwijl ze fysiek gescheiden zijn. De Lower-128 bytes RAM zijn in alle leden van de MCS-51 groep aanwezig. De organisatie ervan is te zien in figuur 7/6.2-7. De laagste 32 bytes zijn gegroepeerd in 4 banken van 8 registers. Door programma instructies worden deze registers opgeroepen als R0 tot en met R7. Door twee bits in het Program Statuswoord (PSW) wordt geselecteerd welke registerbank in gebruik is. Hierdoor kan de coderuimte efficiënter worden gebruikt, omdat registerinstructies korter zijn dan instructies met directe adressering.

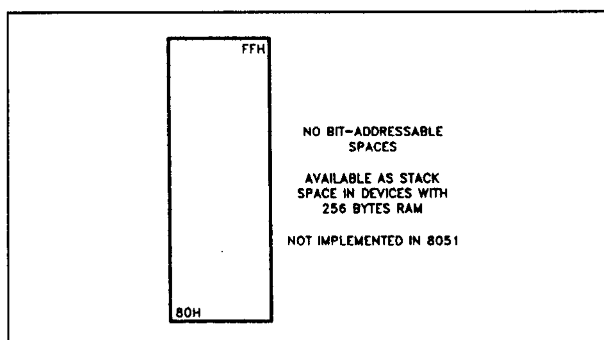
6.2 Algemene eigenschappen van de 8051-familie



Figuur 7/6.2-7: De Lower-128 bytes van de interne RAM.

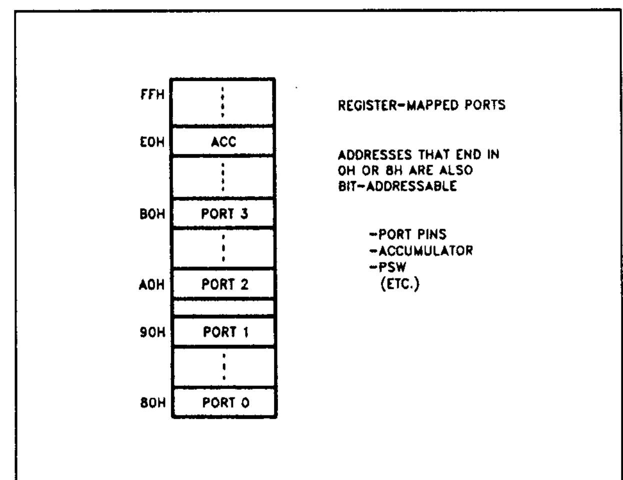
De volgende 16 bytes boven de registerbanken vormen een blok bit-adresseerbare geheugenruimte.

De instructieset van de MCS-51 schakelingen bevat een flink aantal één bit instructies, terwijl de 128 bits in dit gebied door deze instructies direct geadresseerd kunnen worden. De bit adressen in dit gebied lopen van 00H tot en met 7FH. Terwijl de bytes in de Lower-128 zowel direct als indirect toegankelijk zijn, zijn de Upper-128 bytes alleen bereikbaar met indirecte adressering (figuur 7/6.2-8). De Upper-128 bytes zijn niet in de 8051 opgenomen, maar wel in de typen met 256 bytes RAM (8052, enzovoorts).



Figuur 7/6.2-8: De Upper-128 bytes van de interne RAM.

In figuur 7/6.2-9 wordt een kort overzicht gegeven van de Special Function Register (SFR) ruimte. De SFR's omvatten bijvoorbeeld de poortlatches, timers, peripheral controls, enzovoorts. Deze registers zijn alleen bereikbaar met directe adressering. Over het algemeen hebben alle MCS-51 microcontrollers dezelfde SFR's als de 8051 (ook op dezelfde adressen in de SFR-ruimte). Uitgebreider typen van de 8051-familie hebben echter extra SFR's die niet in de 8051 of andere leden van de familie voorkomen. Zestien adressen in de SFR-ruimte zijn zowel bit als byte adresseerbaar.

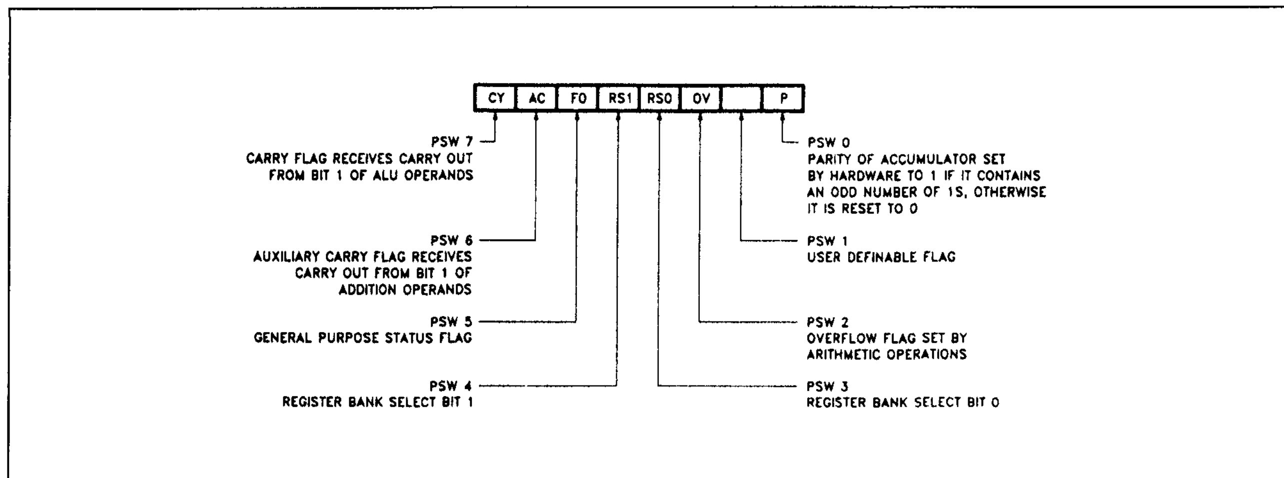


Figuur 7/6.2-9: De SFR-ruimte.

Inleiding op de MCS-51 instructieset

Inleiding

Alle leden van de 8051-familie hebben dezelfde instructieset, die geoptimaliseerd is voor 8 bit besturingstoepassingen. Er zijn verschillende snelle adresseringsmodes voor de toegang tot de interne RAM om byte operaties op kleine datastructuren mogelijk te maken. De instructieset levert uitgebreide ondersteuning van één bit variabelen. Door deze als een apart datatype te beschouwen kunnen bits in besturingen en logische systemen direct worden gemanipuleerd. Hieronder wordt een korte presentatie van de instructieset gegeven.



Figuur 7/6.2-10: Het PSW (Program Status Word) Register.

Programma Statuswoord

Het Program Status Word (PSW) bevat verschillende statusbits die overeenkomen met de toestand van de CPU. Het PSW (zie figuur 7/6.2-10) bevindt zich in de SFR-ruimte. Het bevat het Carry bit, de Auxiliary Carry (voor BCD-operaties), de twee registerbank selectbits, de Overflow flag, een Parity bit en twee statusvlaggen die door de gebruiker kunnen worden gedefinieerd. Het Carry bit dient behalve als carry bit in rekenkundige operaties ook als "Accumulator" voor een aantal Boole'se operaties. De bits RS0 en RS1 worden gebruikt om één van de vier registerbanken uit figuur 7/6.2-7 te selecteren. Een aantal instructies verwijzen naar deze RAM-lokaties als R0 tot en met R7. Het pariteitsbit komt overeen met het aantal 1-en in de Accumulator: P=1 als de Accumulator een oneven aantal enen bevat. Het aantal enen in de Accumulator plus P is dus altijd even. Twee bits in het PSW zijn nog niet toegewezen en kunnen als algemeen toepasbare statusvlag worden gebruikt.

Adresseringsmodes

De adresseringsmodes in de MCS-51 instructieset zijn als volgt:

- **Directe adressering**
  Bij directe adressering wordt de operand gespecificeerd door een 8 bit adresveld in de instructie. Alleen interne data-RAM en SFR's kunnen direct worden geadresseerd.
- **Indirecte adressering**
  Bij indirecte adressering wordt door de instructie een register gespecificeerd dat het adres van de operand bevat. Zowel interne als externe RAM kunnen indirect worden geadresseerd.

Als adresregister voor 8 bit adressen kunnen R0 of R1 van de geselecteerde registerbank of de stackpointer dienen. Als adresregister voor 16 bit adressen komt alleen het 16 bit datapointer register (DPTR) in aanmerking.

Register instructies

De registerbanken die de registers R0 tot en met R7 bevatten, zijn toegankelijk voor sommige instructies die een 3 bit registerspecificatie binnen de opcode hebben. De instructies die op deze wijze de registers bereiken zijn code-efficiënt, omdat hierdoor een adresbyte wordt uitgespaard. Wanneer de instructie wordt uitgevoerd, volgt toegang tot één van de registers in de bank. Met de twee bankselect bits in het PSW wordt tijdens de executie één van de vier banken gekozen.

Registerspecifieke instructies

Sommige instructies zijn specifiek voor een bepaald register.

6.2 Algemene eigenschappen van de 8051-familie

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A − <byte> − C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A − 1	Accumulator only				1
DEC <byte>	<byte> = <byte> − 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

Tabel 7/6.2-2: Overzicht van de rekenkundige instructies van de 8051-familie.

Er zijn bijvoorbeeld instructies die altijd met de Accumulator of de datapointer, enzovoorts werken. Hiervoor is dus geen adresbyte nodig, omdat de opcode dat zelf al doet. Instructies die met A naar de Accumulator verwijzen worden accumulatorspecifieke opcodes genoemd.

Immediate constanten

Na de opcode in het programmageheugen kan de waarde van een constante volgen. Met:
MOV A, #100
wordt de Accumulator bijvoorbeeld geladen met het decimale getal 100. Hetzelfde getal kan ook in hexadecimale digits als 64H worden gespecificeerd.

Geïndexeerde adressering

Met geïndexeerde adressering is alleen het programmageheugen toegankelijk, dat dan ook alleen uitgelezen kan worden. Deze adresseringsmode is bedoeld voor het lezen van opzoektabellen in het programmageheugen. Hierbij wijst een 16 bit basisregister (DPTR of de programcounter) naar het begin van de tabel, terwijl in de Accumulator het tabeltoegangsnummer wordt gezet. Het adres van het begin van de tabel in het programmageheugen wordt gevormd door de accumulatordata op te tellen bij de basispointer.
Een andere soort geïndexeerde adressering wordt gebruikt bij de "case jump" instructie. In dit geval wordt het doeladres van een jumpinstructie berekend als de som van de basispointer en de accumulatordata.

Rekenkundige instructies

Tabel 7/6.2-2 laat zien welke rekenkundige instructies mogelijk zijn. Ook worden de adresseringsmodes getoond die bij elke instructie nodig zijn om de <byte>-operand te bereiken. De ADD A,<byte> instructie kan bijvoorbeeld worden geschreven als:
ADD A,7FH (directe adressering);
ADD A,@R0 (indirecte adressering);
ADD A,R7 (register adressering);
ADD A,#127 (immediate constante).
De executietijden in de tabel gelden als de klokfrequentie 12 MHz is.
Alle rekenkundige instructies worden in 1 μs uitgevoerd, behalve INC DPTR (2 μs) en Multiply en Divide (4 μs). Merk op dat incrementeren of decrementeren van de interne datageheugenruimte mogelijk is zonder door

6.2 Algemene eigenschappen van de 8051-familie

de Accumulator te gaan. Eén van de INC instructies werkt met de 16 bit datapointer. Deze wordt gebruikt om 16 bit adressen voor extern geheugen te genereren.
De MUL AB instructie vermenigvuldigt de Accumulator met de data in het B-register en zet het 16 bit produkt in de aaneengeschakelde B- en accumulatorregisters. De DIV AB instructie deelt de Accumulator door de data in het B-register en laat het 8 bit quotiënt achter in de Accumulator en de 8 bit rest in het B-register. DIV AB wordt in de praktijk vaker gebruikt voor worteltrekken en schuifoperaties dan voor rekenkundige "deel"-opdrachten.
De DA A instructie is voor BCD-bewerkingen. Bij rekenkundige BCD-operaties moeten ADD en ADDC instructies altijd worden gevolgd door een DA A operatie om er zeker van te zijn dat het resultaat ook BCD is. Merk op dat met DA A geen omzetting van binair naar BCD mogelijk is.

Logische instructies

Tabel 7/6.2-3 geeft de lijst met logische instructies van de MCS-51 familie. De instructies die Boole'se operaties (AND, OR, EXOR, NOT) uitvoeren op bytes, doen dat op een bit-na-bit basis. Als de Accumulator en <byte> bijvoorbeeld respectievelijk 00110101B en 01010011B bevatten, zal ANL A,<byte> in de Accumulator 000100001B achterblijven. In deze tabel staan ook de adresseringsmodes die gebruikt kunnen worden om de <byte>-operand te bereiken. De ANL A,<byte> instructie kan bijvoorbeeld één van de volgende vormen hebben:
ANL A,7FH (directe adressering);
ANL A,@R1 (indirecte adressering);
ANL A,R6 (register adressering);
ANL A,#53H (immediate constante).
Alle accumulatorspecifieke instructies worden in 1 µs uitgevoerd (als een 12 MHz clock wordt gebruikt). Voor de overige is 2 µs nodig.

Mnemonic	Operation	Addressing Modes				Execution Time (µs)
		Dir	Ind	Reg	Imm	
ANL A,<byte>	A = A .AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND. A	X				1
ANL <byte>,#data	<byte> = <byte> .AND. #data	X				2
ORL A,<byte>	A = A .OR. <byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR. A	X				1
ORL <byte>,#data	<byte> = <byte> .OR. #data	X				2
XRL A,<byte>	A = A .XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR. A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR. #data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT. A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

Tabel 7/6.2-3: Overzicht van de logische instructies van de 8051-familie.

6.2 Algemene eigenschappen van de 8051-familie

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR,#data16	DPTR = 16-bit immediate constant.				X	2
PUSH <src>	INC SP : MOV "@SP",<src>	X				2
POP <dest>	MOV <dest>,"@SP" : DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

Tabel 7/6.2-4: Datatransfer instructies die toegang geven tot de interne datageheugenruimte.

Merk op dat Boole'se operaties, gebruik makend van directe adressering, op elk willekeurig byte in de Lower-128 intern datageheugenruimte of de SFR-ruimte kunnen worden uitgevoerd, zonder de Accumulator te hoeven gebruiken. De instructie XRL <byte>,#data is bijvoorbeeld een snelle en gemakkelijke manier om poortbits te inverteren, zoals in XRL P1,#0FFH. Als de operatie een gevolg is van een interrupt, bespaart het niet gebruiken van de Accumulator tijd en moeite (om hem in de serviceroutine op te nemen).

De Rotate instructies (RL A, RLC A, enzovoorts) schuiven de inhoud van de Accumulator 1 bit naar links of naar rechts. Door een rotatie naar links rolt het MSB naar de LSB-positie (en voor een rotatie naar rechts andersom: LSB naar MSB). Met de SWAP A instructie worden de hoge en lage nibbles in de Accumulator onderling verwisseld. Dit is nuttig voor BCD-manipulaties. Wanneer de Accumulator bijvoorbeeld een binair getal bevat, waarvan bekend is dat het kleiner dan 100 is, kan het op de volgende manier snel worden omgezet in BCD:

```
MOV B,#10
DIV AB
SWAP A
ADD A,B
```

Door het getal door 10 te delen blijft het decimale cijfer achter in de lage nibble van de Accumulator en het eenhedencijfer in het B-register. De SWAP en ADD instructies verplaatsen het tientallencijfer naar de hoge nibble van de Accumulator en het eenhedencijfer naar de lage nibble.

Data-overdrachten interne RAM

Tabel 7/6.2-4 toont het menu van de instructies die beschikbaar zijn voor het verplaatsen van data binnen de interne adresruimte en de adresseermodes die daarbij gebruikt kunnen worden. Met een 12 MHz clock worden al deze instructies in 1 of 2 μs uitgevoerd.

De MOV <dest>,<scr> instructie maakt het mogelijk om data tussen willekeurige RAM- of SFR-lokaties te verplaatsen, zonder door de Accumulator te gaan. Bedenk dat de Upper-128 bytes van de data-RAM alleen indirect geadresseerd kunnen worden en de SFR-ruimte alleen direct.

Let op dat de stack bij alle 8051-leden in de on-chip RAM huist en naar boven toe aangroeit. De PUSH instructie verhoogt eerst de Stack Pointer met één, waarna het byte naar de stack wordt gekopieerd. PUSH en POP gebruiken alleen directe adressering ter identificatie van het byte dat opgeborgen of teruggehaald wordt, maar de stack wordt zelf bereikt door indirecte adressering met behulp van het SP-register. Dit betekent dat de stack wel in de Upper-128 kan gaan, maar niet in de SFR-ruimte.

6.2 Algemene eigenschappen van de 8051-familie

		2A	2B	2C	2D	2E	ACC
MOV	A,2EH	00	12	34	56	78	78
MOV	2EH,2DH	00	12	34	56	56	78
MOV	2DH,2CH	00	12	34	34	56	78
MOV	2CH,2BH	00	12	12	34	56	78
MOV	2BH,#0	00	00	12	34	56	78

(a) Using direct MOVs: 14 bytes, 9 μs

		2A	2B	2C	2D	2E	ACC
CLR	A	00	12	34	56	78	00
XCH	A,2BH	00	00	34	56	78	12
XCH	A,2CH	00	00	12	56	78	34
XCH	A,2DH	00	00	12	34	78	56
XCH	A,2EH	00	00	12	34	56	78

(b) Using XCHs: 9 bytes, 5 μs

Figuur 7/6.2-11: Het twee plaatsen naar rechts schuiven van een BCD-getal.

Bij microcontrollers die de Upper-128 niet implementeren gaan gePUSHte bytes verloren als de SP naar de Upper-128 verwijst en zijn gePOPte bytes slechts een afgeleide waarde. De Data Transfer instructies bevatten ook een 16 bit MOV die gebruikt kan worden om de Data Pointer (DPTR) te initialiseren voor opzoektabellen in het programmageheugen of voor 16 bit externe datageheugen toegangen.
De XCH A,<byte> instructie maakt dat de Accumulator en het geadresseerde byte data uitwisselen. De XCHD A,@Ri instructie is soortgelijk, maar hierbij zijn alleen de lage nibbles bij de uitwisseling betrokken.
Om te zien hoe XCH en XCHD kunnen worden gebruikt om de datamanipulaties te vergemakkelijken, wordt als voorbeeld het probleem om een 8 cijferig BCD-getal twee plaatsen naar rechts te schuiven genomen. In figuur 7/6.2-11 is te zien hoe dit met behulp van directe MOV's en (ter vergelijking) met XCH instructies gedaan kan worden. Om de werking van de code beter te begrijpen, worden naast iedere instructie de inhouden van de registers die het BCD-getal bevatten en de inhoud van de Accumulator getoond. Nadat de routine is uitgevoerd bevat de Accumulator de twee cijfers die aan de rechter zijde naar buiten werden geschoven. Wanneer directe MOV's voor de routine worden gebruikt, zijn 14 codebytes en 9 μs executietijd nodig (bij 12 MHz). Dezelfde routine wordt met XCH's in bijna de helft van de tijd uitgevoerd, terwijl minder code nodig is.
Voor het naar rechts schuiven van een oneven aantal cijfers moet een één cijfer verschuiving worden uitgevoerd.
In figuur 7/6.2-12 wordt een voorbeeld gegeven van code, gebruik makend van de XCHD-instructie, waardoor een BCD-getal één plaats naar rechts schuift. Ook hierbij zijn de inhouden van de registers en de Accumulator weer vermeld. Eerst worden de pointers R1 en R0 klaargezet om naar de twee bytes te wijzen die de laatste vier BCD-cijfers bevatten. Daarna wordt een programmalus (loop) uitgevoerd waardoor het laatste byte (lokatie 2EH) de laatste twee cijfers van het verschoven getal bevat. De pointers zijn gedecrementeerd en de loop wordt herhaald voor lokatie 2DH. De loop wordt uitgevoerd van LOOP tot CJNE (Compare and Jump if Not Equal) voor R1 = 2EH, 2DH, 2CH en 2BH. Op dat punt is het cijfer dat oorspronkelijk naar rechts werd uitgeschoven op lokatie 2AH terecht gekomen. Omdat op die plaats nullen dienen achter te blijven, wordt het verloren cijfer naar de Accumulator verplaatst.

Data-overdrachten externe RAM

Tabel 7/6.2-5 geeft een overzicht van de Data Transfer instructies die betrekking hebben op extern datageheugen. Hierbij kan alleen indirecte adressering worden gebruikt. Er kan worden gekozen uit een één byte adres (@Ri), waarbij Ri zowel R0 als R1 van de geselecteerde registerbank kan zijn, of een twee byte adres (@DPTR). Het nadeel van het gebruik van 16 bit adressen is (wanneer sprake is van slechts een paar kB externe RAM) dat deze adressen alle 8 bits van poort 2 als adresbus gebruiken. Bij 8 bit adressen is dit niet het geval.
Alle externe RAM-instructies worden in 2 μs uitgevoerd (bij 12 MHz). Merk op dat de Accumulator telkens bron of doel van de data is.

6.2 Algemene eigenschappen van de 8051-familie

Instructie	2A	2B	2C	2D	2E	ACC
MOV R1,#2EH	00	12	34	56	78	XX
MOV R0,#2DH	00	12	34	56	78	XX
loop for R1 = 2EH:						
LOOP: MOV A,@R1	00	12	34	56	78	78
XCHD A,@R0	00	12	34	58	78	76
SWAP A	00	12	34	58	78	67
MOV @R1,A	00	12	34	58	67	67
DEC R1	00	12	34	58	67	67
DEC R0	00	12	34	58	67	67
CJNE R1,#2AH,LOOP						
loop for R1 = 2DH:	00	12	38	45	67	45
loop for R1 = 2CH:	00	18	23	45	67	23
loop for R1 = 2BH:	08	01	23	45	67	01
CLR A	08	01	23	45	67	00
XCH A,2AH	00	01	23	45	67	08

Figuur 7/6.2-12: Het één plaats naar rechts schuiven van een BCD-getal.

Address Width	Mnemonic	Operation	Execution Time (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @DPTR	2

Tabel 7/6.2-5: Datatransfer instructies die toegang geven tot de externe datageheugenruimte.

Mnemonic	Operation	Execution Time (μs)
MOVC A,@A + DPTR	Read Pgm Memory at (A + DPTR)	2
MOVC A,@A + PC	Read Pgm Memory at (A + PC)	2

Tabel 7/6.2-6: Instructies voor het uitlezen van opzoektabellen.

De lees- en schrijfstrobes naar externe RAM worden alleen tijdens het uitvoeren van een MOVX instructie geaktiveerd. Deze signalen zijn normaal niet aktief, zodat de aansluitpennen beschikbaar komen als extra I/O-lijnen als vast staat dat zij helemaal niet voor hun eigenlijke functie zullen worden gebruikt.

Opzoektabellen

Voor het uitlezen van opzoektabellen in het programmageheugen zijn twee instructies beschikbaar (zie tabel 7/6.2-6). Aangezien deze instructies alleen toegang geven tot het programmageheugen kunnen de opzoektabellen niet worden bijgewerkt, maar alleen gelezen. De mnemonic is MOVC (move constante). Als een tabel in extern programmageheugen wordt bereikt, is de leesstrobe $\overline{PSEN}$. De eerste MOVC instructie is voldoende voor een tabel met maximaal 256 ingangen (genummerd van 0 tot 255). Het nummer van de gewenste ingang wordt in de Accumulator gezet, terwijl de Data Pointer naar het begin van de tabel wijst. Daarna kopieert MOVC A,@A+DPTR de gewenste ingang van de tabel naar de Accumulator.
De tweede MOVC instructie werkt net zo, maar hierbij wordt de programcounter (PC) gebruikt als basis van de tabel, terwijl de tabel met behulp van een subroutine wordt bereikt.
Eerst wordt het nummer van de gewenste ingang in de Accumulator geladen:

```
        MOV A,ENTRY_NUMBER
        CALL TABLE
```

De subroutine "TABLE" kan er bijvoorbeeld zo uitzien:

```
TABLE:          MOVC A,@A+PC
                RET
```

De tabel zelf volgt direct na de RET (return) instructie in het programmageheugen. Dit type tabel kan maximaal 255 ingangen hebben (1 tot 255). Nummer 0 kan niet worden gebruikt, omdat op het moment dat de MOVC instructie wordt uitgevoerd, de PC het adres van de RET instructie bevat.

Boole'se instructies

De leden van de 8051-familie bevatten een complete Boole'se (enkel bit) processor. De interne RAM bevat 128 adresseerbare bits en de SFR-ruimte kan maximaal 128 adresseerbare bits ondersteunen. Alle poortlijnen

zijn bit adresseerbaar en kunnen individueel als enkel bit poort worden behandeld. De instructies die toegang geven tot deze bits zijn niet slechts conditionele sprongen (branches), maar een compleet menu van move, set, clear, complement, OR en AND instructies.

Mnemonic	Operation	Execution Time (μs)
ANL C,bit	C = C .AND. bit	2
ANL C,/bit	C = C .AND. .NOT. bit	2
ORL C,bit	C = C .OR. bit	2
ORL C,/bit	C = C .OR. .NOT. bit	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1; CLR bit	2

Tabel 7/6.2-7: Overzicht van de Boole'se instructies van de MCS-51 familie.

De instructieset voor de Boole'se processor is te zien in tabel 7/6.2-7. Alle bits kunnen alleen worden bereikt met directe adressering. De bitadressen 00H tot en met 7FH bevinden zich in de Lower-128, terwijl de adressen 80H tot en met FFH zich in de SFR-ruimte bevinden. Dit maakt het bijvoorbeeld gemakkelijk om een interne vlag naar een poortpen te verplaatsen:

```
MOV   C,FLAG
MOV   P1.0,C
```

In dit voorbeeld is FLAG de naam van een willekeurig adresseerbaar bit in de Lower-128 of SFR-ruimte Een I/O-lijn (in dit geval de LSB van poort 1) wordt geset of gecleared, afhankelijk van de vlagbit die 1 of 0 kan zijn.

Het carrybit in het PSW wordt gebruikt als de enkele bit accumulator van de Boole'se processor.
Bitinstructies die met een C naar het carrybit verwijzen worden Carry-specifieke instructies genoemd (bijvoorbeeld CLR C). Het carrybit heeft ook een direct adres, aangezien het is opgenomen in het PSW-register dat bit adresseerbaar is. Merk op dat de Boole'se instructieset wel ANL en ORL operaties bevat, maar geen XRL (Exclusive OR). Een XRL operatie kan gemakkelijk met software worden uitgevoerd. Veronderstel bijvoorbeeld dat de Exclusive OR-functie van twee bits nodig is:
C = bit1 .XRL. bit2
Dit kan worden uitgevoerd met:

```
        MOV   C,bit1
        JNB   bit2,OVER
        CPL   C
OVER:         (continue).
```

Eerst wordt bit1 naar de carry overgebracht. Als bit2 = 0, dan bevat C het juiste resultaat (bit1 .XRL. bit2 = bit1 als bit2 = 0). Als bit2 echter = 1 was bevat C het complement van het resultaat, zodat het alleen nog geïnverteerd hoeft te worden (CPL C) om de operatie te beëindigen. Hier werd de JNB instructie gebruikt, die een jump over de CPL C instructie uitvoert als het geadresseerde bit2 niet is geset.

Relatieve offset
Het doeladres voor de sprongen (jumps) wordt in de assembler gespecificeerd door een label of door een echt adres in het programmageheugen. Het doeladres assembleert echter tot een relatieve offsetbyte. Dit is een offsetbyte met teken (2's complement) dat wordt opgeteld bij de PC. Het bereik van de jump bedraagt daarom -128 tot +127 programmageheugenbytes ten opzichte van het eerste byte na de instructie.

Jump instructies
Tabel 7/6.2-8 is een lijst van onvoorwaardelijke jumps. In deze tabel is één "JMP addr" instructie te zien, maar in werkelijkheid zijn

er drie: SJMP, LJMP en AJMP die onderling verschillen door het formaat van het doeladres. JMP is een generisch mnemonic dat kan worden gebruikt als het de programmeur niet kan schelen hoe de jump wordt geëncodeerd.

Mnemonic	Operation	Execution Time (µs)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

Tabel 7/6.2-8: Onvoorwaardelijke jumps in MCS-51 schakelingen.

De SJMP instructie encodeert het doeladres als een relatieve offset. De instructie is 2 bytes lang en bestaat uit de opcode en het relatieve offsetbyte. De sprongafstand wordt begrensd tot -128 à +127 bytes ten opzichte van de instructie die op de SJMP volgt.
De LJMP instructie encodeert het doeladres als een 16 bit constante. Deze instructie is 3 bytes lang, omdat hij uit de opcode en twee adresbytes bestaat.
Het doeladres kan zodoende overal in de 64 kB programmageheugenruimte liggen.
De AMJP instructie tenslotte encodeert het doeladres als een 11 bit constante. De instructie is 2 bytes lang en bestaat uit de opcode die zelf 3 van de 11 adresbits bevat, gevolgd door nog een byte met de lage 8 bits van het doeladres. Wanneer de instructie wordt uitgevoerd, vervangen deze 11 bits gewoon de lage 11 bits in de PC. De hoge 5 bits blijven hetzelfde, zodat het doel binnen hetzelfde 2 kB blok van de instructie na de AJMP moet liggen. In alle gevallen specificeert de programmeur het doeladres naar de assembler op dezelfde manier: als een label of als een 16 bit constante. De assembler zet het doeladres om in het juiste formaat voor de betreffende instructie. Als het formaat dat voor de instructie nodig is de afstand van het gespecificeerde doeladres niet ondersteunt wordt een "destination out of range" bericht in de List-file geschreven.
De JMP @A+DPTR-instructie ondersteunt "case jumps". Het doeladres wordt tijdens de uitvoering berekend als de som van het 16 bit DPTR-register en de Accumulator. Meestal bevat DPTR het adres van een jumptabel, terwijl de Accumulator een index naar de tabel krijgt. Bij een 5 way branch wordt bijvoorbeeld een integer 0 tot 4 in de Accumulator geladen. De uit te voeren code kan er dan als volgt uitzien:

```
        MOV   DPTR,#JUMP_TABLE
        MOV   A,INDEX_NUMBER
        RL    A
        JMP   @A_DPTR
```

De RL A-instructie zet het indexgetal (0 tot 4) om in een even getal tussen 0 en 8, omdat elke toegang tot de jumptabel 2 bytes lang is:

```
JUMP_TABLE:
        AJMP  CASE_0
        AJMP  CASE_1
        AJMP  CASE_2
        AJMP  CASE_3
        AJMP  CASE_4
```

In tabel 7/6.2-8 is ook een enkele "CALL addr" instructie te zien, waar er in werkelijkheid twee van zijn: LCALL en ACALL. Deze hebben een verschillend formaat voor het subroutine-adres dat naar de CPU gaat. CALL is een generisch mnemonic voor het geval de programmeur niet weet hoe het adres wordt geëncodeerd.
De LCALL instructie gebruikt het 16 bit adresformaat en de subroutine kan overal in de 64 kB programmageheugenruimte liggen.
De ACALL instructie maakt gebruik van het 11 bit formaat, zodat de subroutine in hetzelfde 2 kB blok moet liggen als de instructie die op ACALL volgt.
In beide gevallen specificeert de programmeur het subroutine-adres naar de assembler op dezelfde manier: als een label of als een 16 bit constante.

6.2 Algemene eigenschappen van de 8051-familie

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

Tabel 7/6.2-9: Voorwaardelijke jumps bij MCS-51 familieleden.

De assembler zet het adres in het juiste formaat voor de betreffende instructies.
Subroutines moeten eindigen met een RET instructie om de uitvoering over te geven aan de instructie die na de CALL komt.
RETI wordt gebruikt om terug te keren vanuit een serviceroutine. Het enige verschil tussen RET en RETI is dat RETI aan het interrupt besturingssysteem vertelt dat de lopende interruptie afgehandeld is. Als er geen interrupt loopt op het moment dat RETI wordt uitgevoerd, is RETI functioneel gelijk aan RET.
Tabel 7/6.2-9 geeft een lijst van voorwaardelijke jumps die de MCS-51 gebruiker ter beschikking staan. Bij al deze jumps wordt het doeladres gespecificeerd met de relatieve offsetmethode, zodat de sprongafstand beperkt is tot -128 à +127 bytes vanaf de instructie die volgt op de voorwaardelijke jump. Het is echter belangrijk om te weten dat de gebruiker het feitelijke doeladres naar de assembler op dezelfde wijze specificeert als bij de andere jumps: als een label of als een 16 bit constante.
In het PSW is geen Zero-bit, zodat de JZ en JNZ instructies die conditie in de Accumulatordata testen.
De DJNZ instructie (Decrement and Jump if Not Zero) dient voor lusbesturingen. Om een lus (loop) n maal uit te voeren wordt een tellerbyte met n geladen, terwijl de loop met DJNZ naar het begin van de loop eindigt. In het volgende voorbeeld is n = 10:

```
         MOV    TELLER,#10
LOOP:    (begin loop)
           *
           *
           *
         (einde loop)
         DJNZ   TELLER,LOOP
      (continue)
```

De CJNE instructie (Compare and Jump if Not Equal) kan ook worden gebruikt voor lusbesturing zoals in figuur 7/6.2-12. In het operand field van de instructie worden twee bytes gespecificeerd. De jump wordt nu alleen uitgevoerd als de twee bytes niet gelijk zijn. In het voorbeeld van figuur 7/6.2-12 waren de twee bytes de data in R1 en de constante 2AH. In het begin was de data in R1 2EH. Iedere keer dat de lus werd doorlopen, werd R1 met één verlaagd totdat 2AH werd bereikt.
Deze instructie kan ook voor "groter dan" en "kleiner dan" vergelijkingen worden gebruikt. De twee bytes in het operand field worden dan beschouwd als integers zonder teken. Als de eerste kleiner is dan de tweede wordt de carrybit gezet, terwijl in het omgekeerde geval de carrybit wordt gecleared.

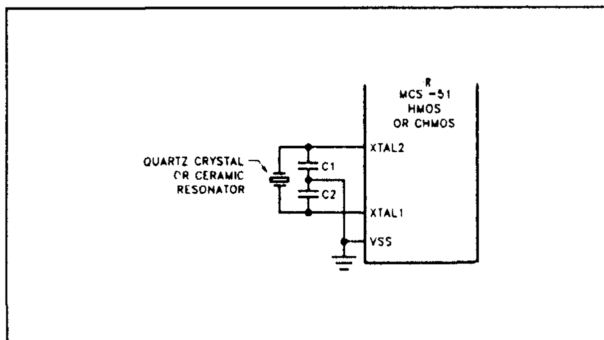
Overige hardware gegevens

Interne clock

Alle microcontrollers uit de MCS-51 serie hebben een oscillator op de chip die desgewenst als clock voor de CPU kan worden gebruikt. De interne clockgenerator bepaalt de volgorde van toestanden waaruit de

6.2 Algemene eigenschappen van de 8051-familie

MCS-51 machinecyclus is opgebouwd. Om de on-chip oscillator te kunnen gebruiken moet men een kristal of een ceramische resonator tussen de pennen XTAL1 en XTAL2 van de microcontroller aansluiten, met condensatoren naar aarde zoals in figuur 7/6.2-13 wordt getoond.



Figuur 7/6.2-13: Toepassing van de on-chip oscillator.

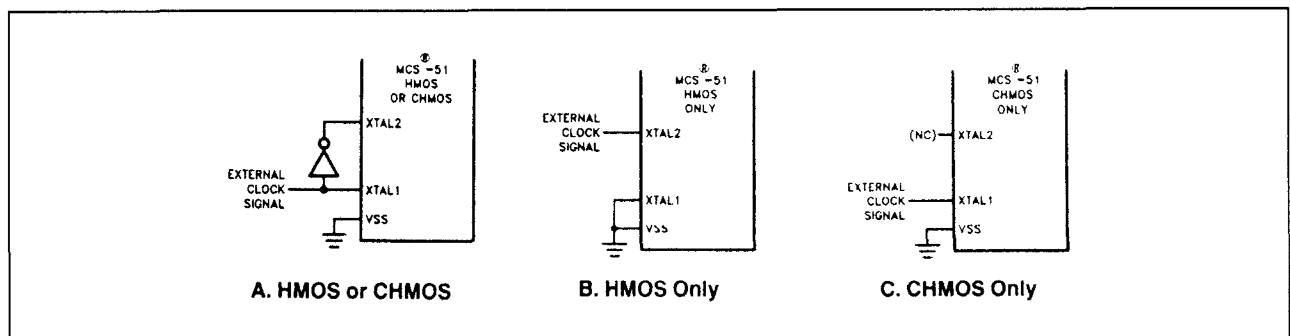
Externe clock

In figuur 7/6.2-14 is te zien hoe de clock kan worden aangestuurd door een externe oscillator. Let op dat bij HMOS-schakelingen zoals de “gewone” 8051 de interne clockgenerator wordt aangedreven op de XTAL2-pen. Bij CHMOS-processoren (bijvoorbeeld de 80C51) gebeurt dat op de XTAL1-pen. Men dient dus voorzichtig te zijn.

Machinecycli

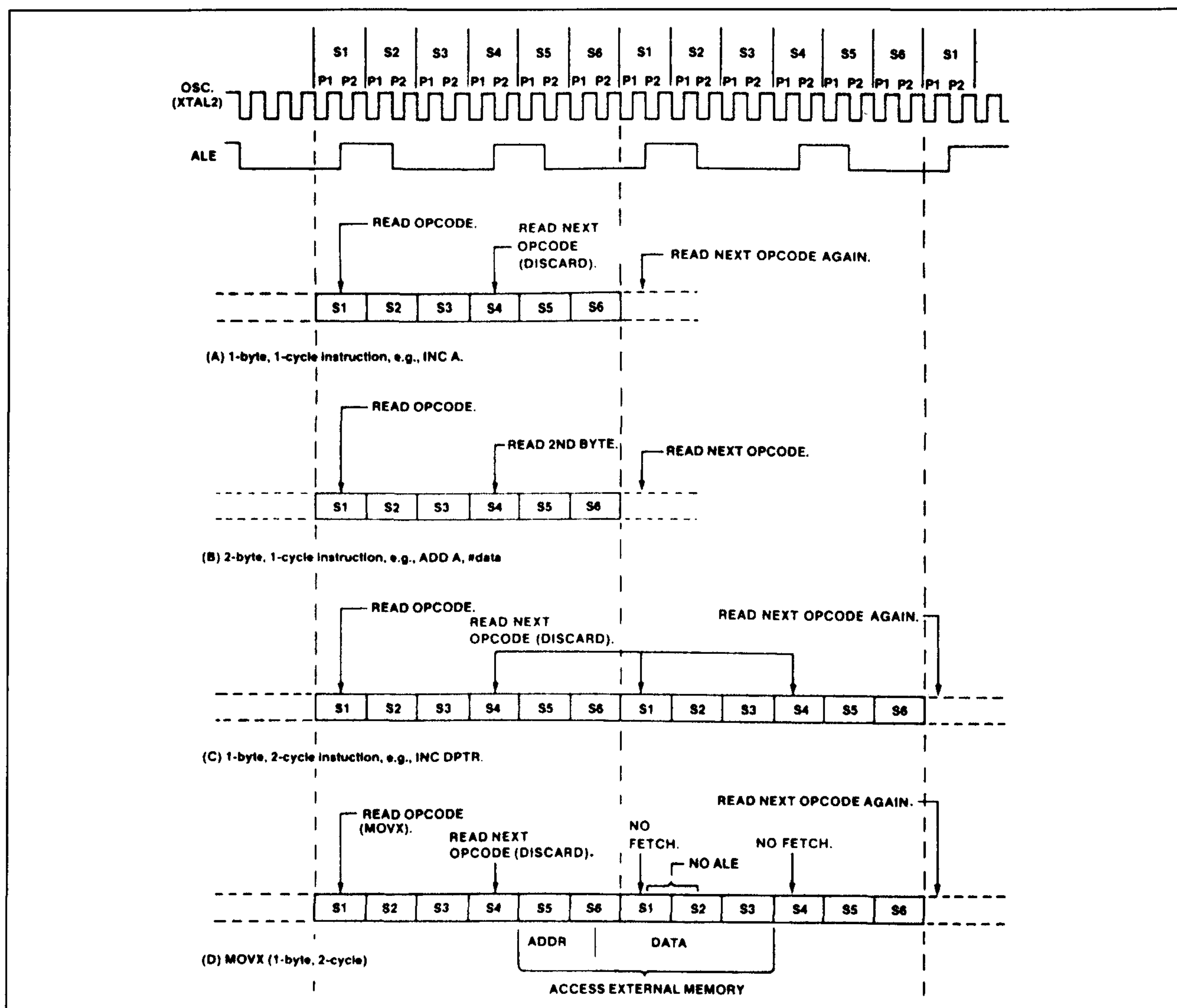
Een machinecyclus bestaat uit een opeenvolging van 6 toestanden, genummerd van S1 tot en met S6. Elke toestand duurt twee oscillatorperioden. Een complete machinecyclus duurt dus 12 oscillatorperioden van 1 μs als de oscillatiefrequentie 12 MHz bedraagt. Elke toestand wordt verdeeld in een Phase 1-helft en een Phase 2-helft. In figuur 7/6.2-15 zijn bijvoorbeeld de fetch/execute-volgorden te zien in toestanden en fasen voor verschillende soorten instructies. Gewoonlijk worden tijdens iedere machinecyclus twee programfetches gegenereerd, ook als de uit te voeren instructie die niet nodig heeft: in dat geval negeert de CPU de extra fetch gewoon en wordt de Program Counter niet verhoogd. De uitvoering van een één cyclus instructie (figuur 7/6.2-15A en -B) begint tijdens toestand 1 van de machinecyclus wanneer de opcode in het Instructie Register wordt gelatcht. Tijdens S4 van dezelfde machinecyclus treedt een tweede fetch op. De executie is klaar op het einde van toestand 6 van deze machinecyclus.

De MOVX instructies (figuur 7/6.2-15D) hebben twee machinecycli nodig. Tijdens de tweede cyclus van een MOVX instructie wordt geen programfetch uitgevoerd. Dit is de enige keer dat programfetches worden overgeslagen. De fetch/execute-volgorden zijn gelijk bij intern of extern programmageheugen, terwijl ook de executietijden niet van intern of extern geheugen afhankelijk zijn. Figuur 7/6.2-16 toont de signalen en de timing die betrekking hebben op programfetches bij extern programmageheugen. In dat geval wordt de Program Memory read strobe $\overline{PSEN}$ tweemaal per machinecyclus geaktiveerd, hetgeen te zien is in figuur 7/6.2-16A.



Figuur 7/6.2-14: Aansluitingen van een externe clock. Let op het verschil tussen CMOS en HMOS.

6.2 Algemene eigenschappen van de 8051-familie

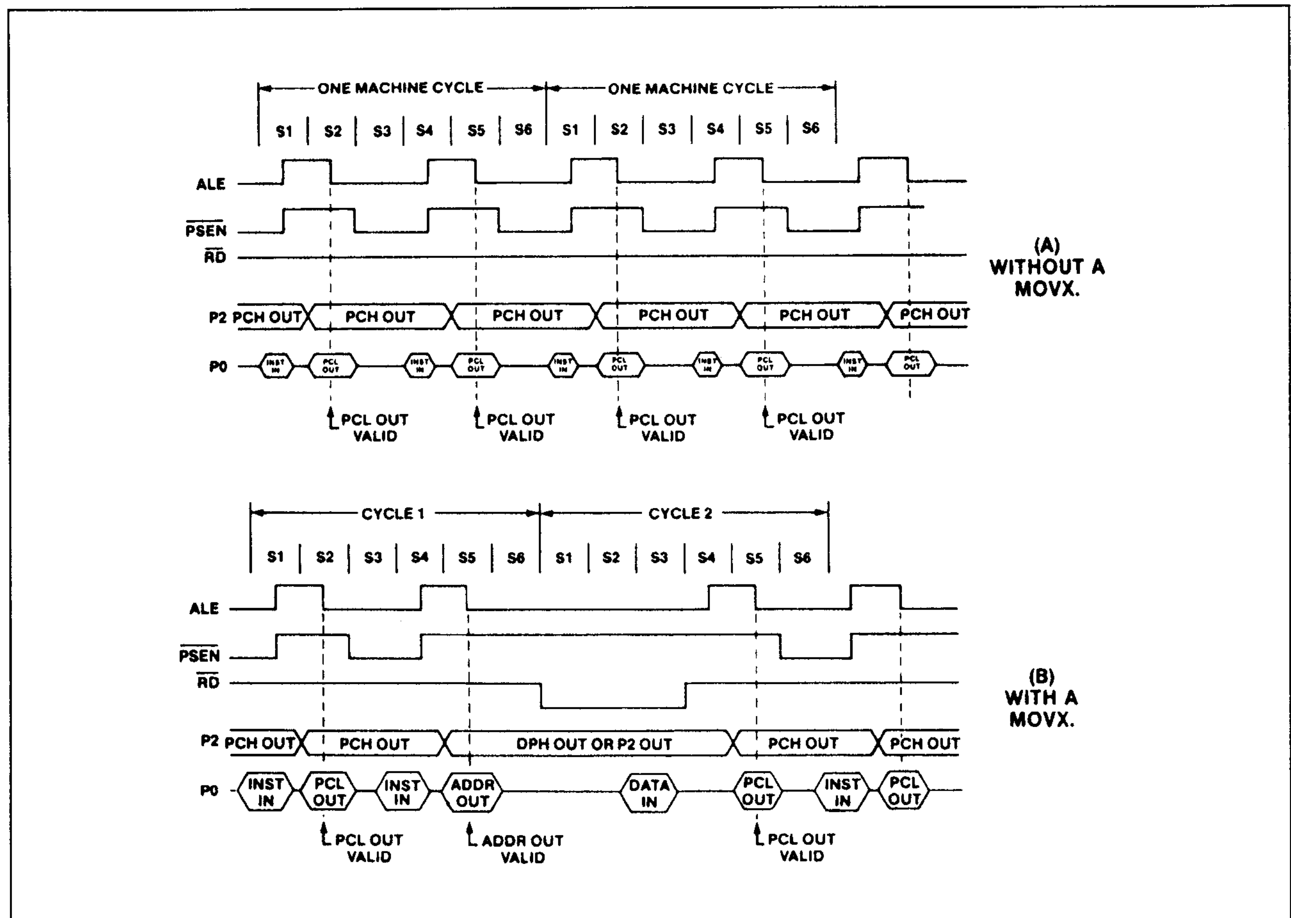


Figuur 7/6.2-15: Toestand volgorden in MCS-51 processoren.

Wanneer een toegang tot extern datageheugen optreedt (zie figuur 7/6.2-16B) worden twee $\overline{\text{PSEN}}$'s overgeslagen, omdat de adres- en databus hiervoor worden gebruikt. Het is duidelijk dat een datageheugen buscyclus tweemaal zo lang duurt als een programmageheugen buscyclus.
In figuur 7/6.2-16 is de relatieve timing van de adressen te zien die worden verzonden via poort 0 en poort 2 en van ALE en $\overline{\text{PSEN}}$.

ALE wordt gebruikt om het laag adresbyte van P0 in de adreslatch op te slaan. Wanneer de CPU met intern programmageheugen werkt, wordt $\overline{\text{PSEN}}$ niet geaktiveerd en worden geen programma-adressen verzonden. ALE blijft echter tweemaal per machinecyclus aktief en is beschikbaar als clockuitgangssignaal. Let echter op dat tijdens de uitvoering van de MOVX instructie één ALE wordt overgeslagen.

6.2 Algemene eigenschappen van de 8051-familie



Figuur 7/6.2-16: Buscycli bij MCS-51 processoren, waarbij vanuit extern programmageheugen wordt geëxecuteerd.

Interruptstructuur

Vijf interruptbronnen

De kern van de 8051 verzorgt vijf interruptiebronnen: 2 externe interrupts, 2 timer interrupts en de interrupt van de seriële poort. Hierna volgt een overzicht van de interruptstructuur voor de 8051. Andere leden van deze familie hebben extra interruptbronnen en -vectoren, die ter plaatse worden behandeld.

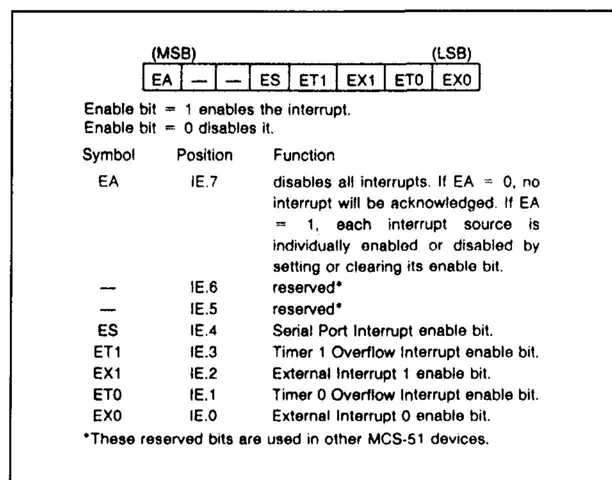
Interrupt enables

Elk van de interruptbronnen kan individueel worden vrijgegeven of gesperd (enabled, disabled) door setten of clearen van een bit in het Interrupt Enable-register (IE: figuur 7/6.2-17) van de SFR. Dit register bevat ook een globaal disablebit waarmee alle interrupts in één keer kunnen worden gesperd.

Interrupt prioriteiten

Aan elke interruptiebron kan ook individueel een prioriteitsniveau (1 uit 2) worden toegewezen door een bit in het Interrupt Priority (IP) register van de SFR te setten of te clearen. Figuur 7/6.2-18 laat het IP register van de 8051 zien. Een interrupt met een lage prioriteit kan wel worden onderbroken door een interrupt met een hogere prioriteit, maar niet door een met dezelfde lage prioriteit. Een interrupt met hoge prioriteit kan niet worden onderbroken door een andere interruptiebron.

6.2 Algemene eigenschappen van de 8051-familie



(MSB) (LSB)

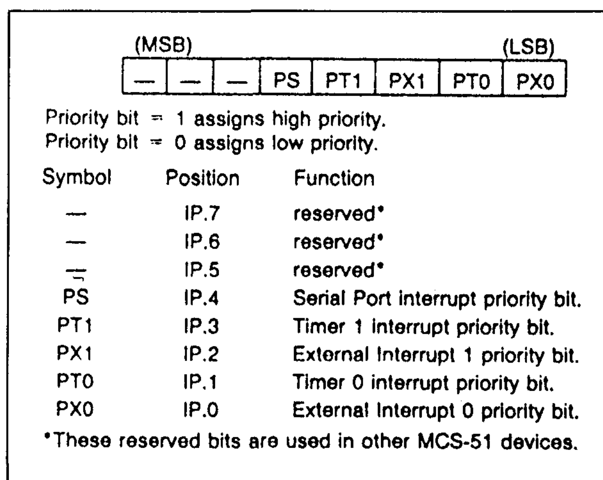
EA	—	—	ES	ET1	EX1	ET0	EX0

Enable bit = 1 enables the interrupt.
Enable bit = 0 disables it.

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved*
—	IE.5	reserved*
ES	IE.4	Serial Port Interrupt enable bit.
ET1	IE.3	Timer 1 Overflow Interrupt enable bit.
EX1	IE.2	External Interrupt 1 enable bit.
ET0	IE.1	Timer 0 Overflow Interrupt enable bit.
EX0	IE.0	External Interrupt 0 enable bit.

*These reserved bits are used in other MCS-51 devices.

Figuur 7/6.2-17: Interrupt Enable Register (IE) van de 8051.



(MSB) (LSB)

—	—	—	PS	PT1	PX1	PT0	PX0

Priority bit = 1 assigns high priority.
Priority bit = 0 assigns low priority.

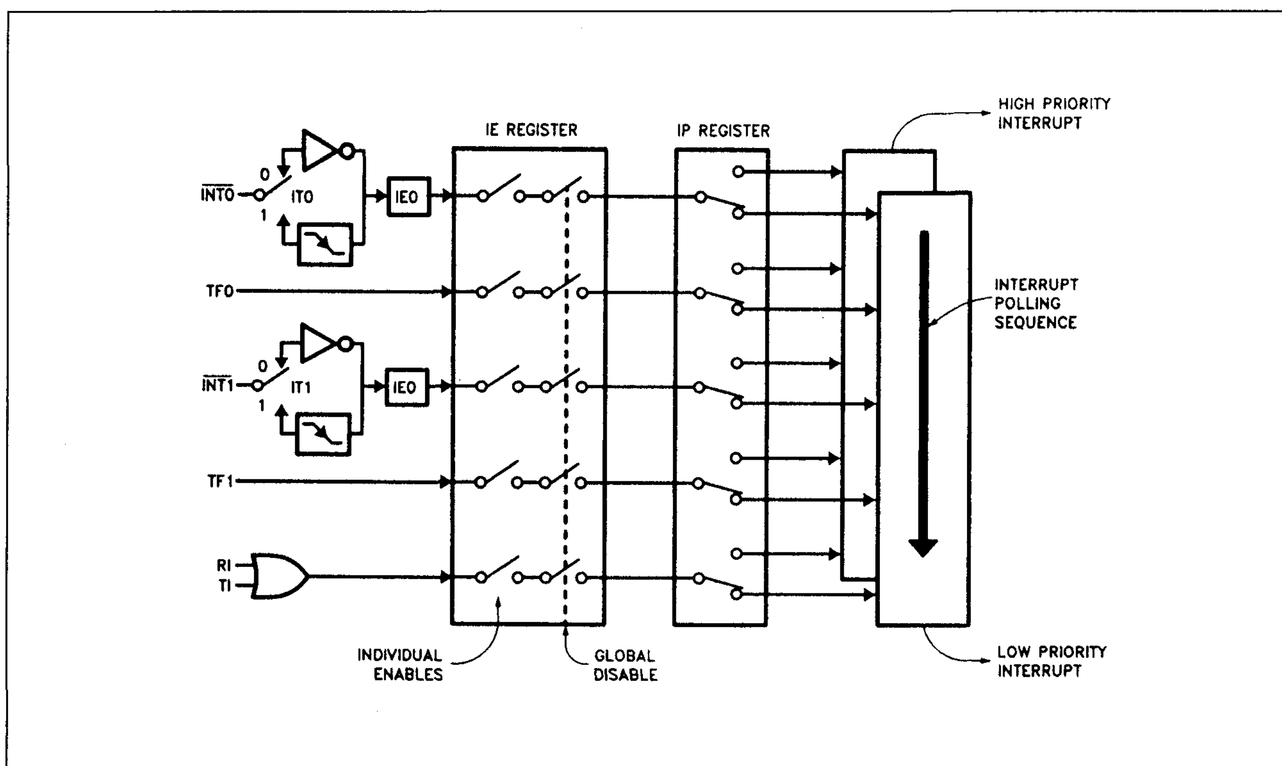
Symbol	Position	Function
—	IP.7	reserved*
—	IP.6	reserved*
—	IP.5	reserved*
PS	IP.4	Serial Port interrupt priority bit.
PT1	IP.3	Timer 1 interrupt priority bit.
PX1	IP.2	External interrupt 1 priority bit.
PT0	IP.1	Timer 0 interrupt priority bit.
PX0	IP.0	External interrupt 0 priority bit.

*These reserved bits are used in other MCS-51 devices.

Figuur 7/6.2-18: Interrupt Priority Register (IP) van de 8051.

Als twee interruptrequests met verschillende prioriteitsniveaus tegelijkertijd worden ontvangen, wordt het request met het hoogste niveau behandeld. Als twee gelijktijdig ontvangen interruptrequests hetzelfde prioriteitsniveau hebben, bepaalt interne polling welk request wordt behandeld.

Binnen elk prioriteitsniveau is er dus een tweede prioriteitsstructuur die bepaald wordt door de pollingvolgorde. In figuur 7/6.2-19 is voor de 8051 te zien hoe de IE en IP registers plus de pollingvolgorde werken om te bepalen welke interrupt behandeld zal worden.



Figuur 7/6.2-19: Het 8051 interrupt besturingssysteem.

6.2 Algemene eigenschappen van de 8051-familie

In bedrijf worden alle interruptvlaggen tijdens toestand 5 van iedere machinecyclus naar het interrupt besturingssysteem gelatcht. De monsters worden tijdens de volgende machinecyclus afgevraagd. Als de vlag van een vrijgegeven interrupt geset (1) blijkt te zijn, genereert het interruptsysteem een LCALL naar de betreffende lokatie in het programmageheugen, tenzij de een of andere conditie de interrupt blokkeert. Een interrupt kan door verschillende condities worden tegen gehouden, waaronder een interrupt van hetzelfde of een hoger prioriteitsniveau die al bezig is. De met hardware gegenereerde LCALL maakt dat de inhoud van de Program Counter de stack wordt opgeduwd en dat de PC wordt geladen met het beginadres van de serviceroutine. Zoals reeds werd gezegd bij figuur 7/6.2-3 begint de serviceroutine voor elke interrupt bij een vaste lokatie.
Alleen de Program Counter wordt automatisch op de stack gezet (niet het PSW of een ander register). Hierdoor kan de programmeur zelf beslissen hoeveel tijd besteed kan worden aan het opslaan van andere registers. Als gevolg hiervan kunnen veel interruptfuncties die specifiek zijn voor besturingen, zoals het omzetten van een poortpen, het herladen van een timer of het leeghalen van een seriële buffer, in minder tijd worden uitgevoerd.

Simulatie van een derde prioriteitsniveau

Bij sommige toepassingen zijn meer prioriteitsniveaus nodig dan de twee waarin wordt voorzien door de hardware van de MCS-51 schakelingen. In dergelijke gevallen kan betrekkelijk eenvoudige software worden geschreven om hetzelfde effect te bereiken als een derde prioriteitsniveau. Eerst worden interrupts die een hogere prioriteit dan 1 moeten hebben toegewezen aan prioriteit 1 in het IP (Interrupt Priority) register. De serviceroutines voor interrupts met prioriteit 1, die onderbreekbaar moeten zijn door interrupts met "prioriteit 2" kunnen de volgende code bevatten:

```
        PUSH  IE
        MOV   IE,#MASK
        CALL  LABEL
              *******
        (voer service-routine uit)
              *******
        POP   IE
        RET
LABEL:  RETI
```

Zodra een interrupt met prioriteit 1 wordt bevestigd, wordt het IE-register opnieuw gedefinieerd om alle interrupts, behalve "prioriteit 2" te sperren. Daarna voert een CALL naar LABEL de RETI instructie uit, waardoor de prioriteit 1 interrupt-in-behandeling flipflop wordt gecleared. Op dit punt kan iedere interrupt met prioriteit 1 die is vrijgegeven worden bediend, maar zijn verder alleen "prioriteit 2" interrupts vrijgegeven.
Door IE te POPpen wordt de originele byte teruggezet. Daarna wordt een gewone RET gebruikt om de serviceroutine te beëindigen. De extra software voegt 10 µs toe aan de prioriteit 1 interrupts (bij 12 MHz).

Programmeer-handleiding en instructieset

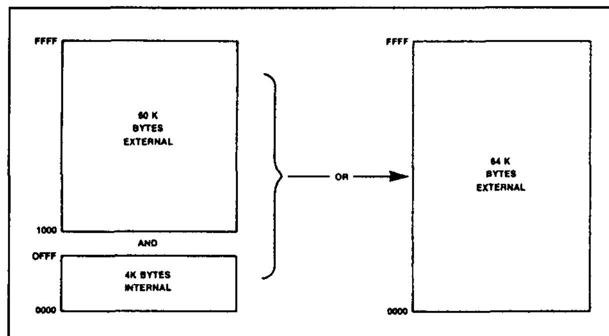
Inleiding

Dit gedeelte is toegesneden op het programmeren van de leden van de MCS-51 familie. De nadruk ligt hierbij op de 8051, de 8052 en de 80C51. De extra programmeereigenschappen van andere typen worden bij de beschrijving daarvan behandeld. De geheugens en registers zijn in dit gedeelte iets anders weergegeven dan in het voorgaande om het programmeren eenvoudiger te maken.

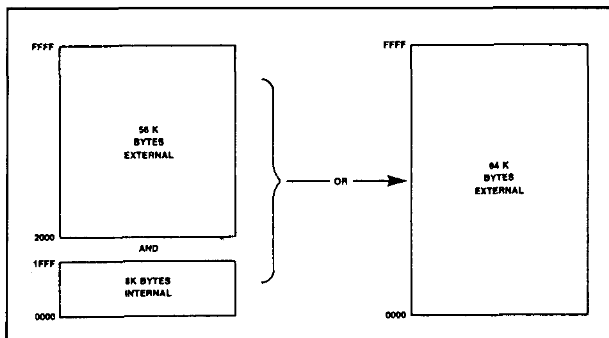
Organisatie programmageheugen

De 8051 heeft aparte adresruimten voor programmageheugen en datageheugen. Het programmageheugen kan 64 kB lang zijn, waarvan de laagste 4 kB (8 kB voor de 8052) zich op de chip kan bevinden.

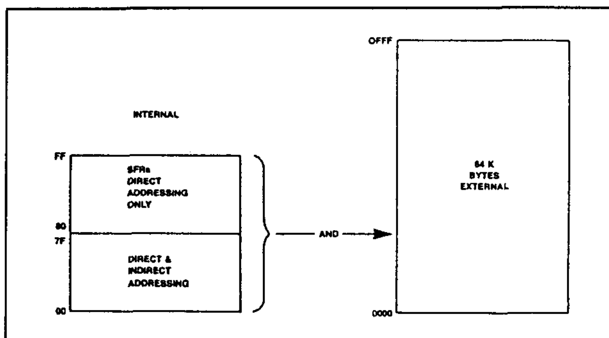
6.2 Algemene eigenschappen van de 8051-familie



Figuur 7/6.2-20: Programmageheugen van de 8051 microcontroller.



Figuur 7/6.2-21: Programmageheugen van de 8052.



Figuur 7/6.2-22: Datageheugen van de 8051.

Figuur 7/6.2-20 laat de indeling van het programmageheugen van de 8051 zien en figuur 7/6.2-21 die van de 8052.

Organisatie datageheugen

De 8051 kan maximaal 64 kB datageheugen buiten de chip adresseren.

Met MOVX wordt toegang verkregen tot extern datageheugen. De 8051 heeft 128 bytes on-chip RAM (de 8052 heeft 256 bytes) plus een aantal Special Function Registers (SFR's).

De Lower-128 RAM kunnen zowel met directe adressering (:MOV data adres) of met indirecte adressering (:MOV @Ri) worden bereikt. In figuur 7/6.2-22 is de organisatie van het datageheugen van de 8051 te zien en in figuur 7/6.2-23 die van de 8052.

Indirecte adresruimte

Let op dat in figuur 7/6.2-22 de SFR's en de indirecte adres RAM's dezelfde adressen hebben (80H tot en met 0FFH). Ondanks dat zijn het twee aparte gebieden die op twee verschillende manieren worden bereikt.

De instructie:

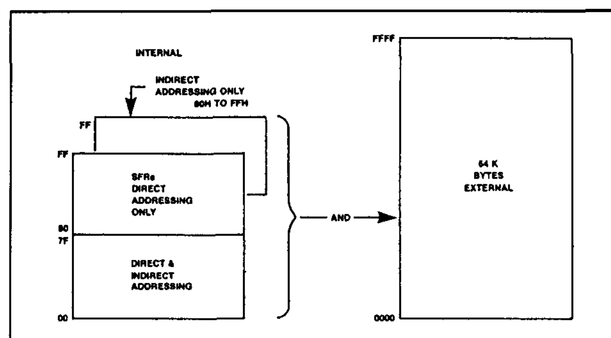
```
MOV   80H,#0AAH
```

schrijft bijvoorbeeld 0AAH naar poort 0 (die één van de SFR's is) en de instructie:

```
MOV   R0,#80H
MOV   @R0,#0BBH
```

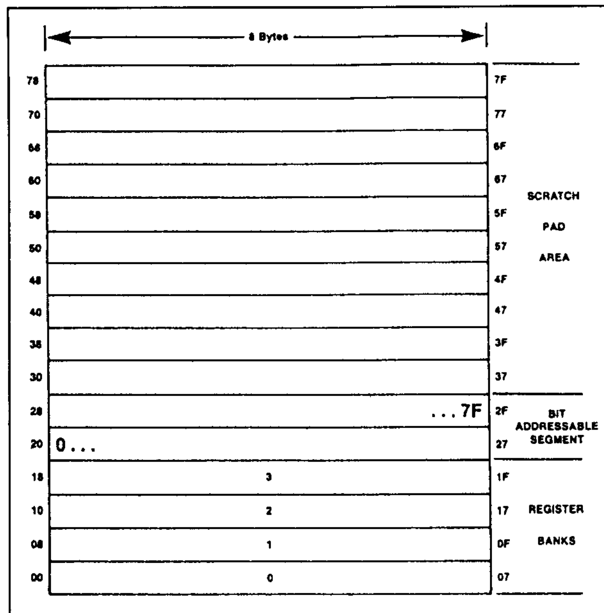
schrijft 0BBH in lokatie 80H van de data-RAM. Na uitvoering van beide bovenstaande instructies zal poort 0 dus 0AAH bevatten en lokatie 80 van de RAM 0BBH.

Merk op dat de stackoperaties voorbeelden van indirecte adressering zijn. De Upper-128 bytes data-RAM zijn dus beschikbaar als stackruimte voor die controllers die 256 bytes interne RAM implementeren.



Figuur 7/6.2-23: Datageheugen van de 8052 microcontroller.

6.2 Algemene eigenschappen van de 8051-familie



Figuur 7/6.2-24: 128 bytes direct en indirect adresseerbare RAM.

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*+T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

* = Bit addressable
+ = 8052 only

Tabel 7/6.2-10: Alle SFR's en hun adressen.

Directe en indirecte adresruimten
De 128 bytes RAM die zowel met directe als indirecte adressering bereikbaar zijn, kunnen worden onderverdeeld in 3 segmenten (figuur 7/6.2-24).

- **Registerbanken 0 tot en met 3**
  Lokaties 0 tot en met 1FH (32 bytes). ASM-51 en de schakeling na reset wijzen automatisch naar registerbank 0. Om de andere registerbanken te gebruiken moet de programmeur ze in de software selecteren.
  Elke bank bevat 8 één byte registers (0 tot en met 7). Reset zet de Stack Pointer op lokatie 07H die met één wordt verhoogd om te beginnen bij lokatie 08H (die het eerste register R0 is van de tweede registerbank). Om dus meer dan één registerbank te kunnen gebruiken, moet de SP naar een andere lokatie in de RAM worden geïnitialiseerd die niet voor dataopslag wordt gebruikt (dat wil zeggen: een hoger gedeelte van de RAM).
- **Bit adresseerbare gebieden**
  Voor dit segment zijn 16 bytes toegewezen (20H tot en met 2FH).
  Elk van de 128 bits van dit segment kan direct worden geadresseerd (0 tot en met 7FH). Naar de bits kan op twee manieren (die beide acceptabel zijn voor de ASM-51) worden verwezen. De ene manier is te verwijzen naar hun adressen (dus 0 tot en met 7FH). De andere manier is te refereren aan de bytes 20H tot en met 2FH.
  De bits 0 t/m 7 kunnen dus ook bits 20.0 tot en met 20.7 worden genoemd en de bits 8 tot en met FH zijn dan 21.0 tot en met 21.7, enzovoorts. Elk van de 16 bytes in dit segment kan ook als byte worden geadresseerd.
- **Scratch Pad gebied**
  Bytes 30H tot en met 7FH staan de gebruiker ter beschikking als data-RAM. Als de stack pointer echter is toegewezen aan dit gebied dienen voldoende bytes over gehouden te worden om SP dataverlies te voorkomen.

6.2 Algemene eigenschappen van de 8051-familie

8 Bytes

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

↑ Bit Addressable

Figuur 7/6.2-25: Adresindeling (memory map) van de SFR's.

Speciale functieregisters

Tabel 7/6.2-10 geeft een overzicht van alle SFR's en de adressen daarvan. Wanneer tabel 7/6.2-10 wordt vergeleken met figuur 7/6.2-25 blijkt dat alle SFR's die byte en bit adresseerbaar zijn, zich in de eerste kolom van figuur 7/6.2-25 bevinden. Tabel 7/6.2-11 laat zien wat er in de SFR's staat direct na het resetten (en na het opkomen van de voedingsspanning). Hierna wordt aandacht besteed aan SFR's waarvan de bits zijn toegewezen aan verschillende functies.

Interrupts

Om een interrupt te kunnen gebruiken moeten de volgende drie stappen worden doorlopen:

- Zet de EA-bit (Enable All) in het IE op 1.
- Zet de bedoelde individuele interrupt enablebit in het IE-register op 1.
- Begin de interrupt serviceroutine op het overeenkomstige vectoradres van die interrupt (zie tabel 7/6.2-12).

Register	Value in Binary
*ACC	00000000
*B	00000000
*PSW	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
*P0	11111111
*P1	11111111
*P2	11111111
*P3	11111111
*IP	8051 XXX00000, 8052 XX000000
*IE	8051 0XX00000, 8052 0X000000
TMOD	00000000
*TCON	00000000
*+T2CON	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
+TH2	00000000
+TL2	00000000
+RCAP2H	00000000
+RCAP2L	00000000
*SCON	00000000
SBUF	Indeterminate
PCON	HMOS 0XXXXXXX CHMOS 0XXX0000

X = Undefined
* = Bit Addressable
+ = 8052 only

Tabel 7/6.2-11: Inhoud van de SFR's na het resetten.

6.2 Algemene eigenschappen van de 8051-familie

CY	AC	F0	RS1	RS0	OV	—	P

CY	PSW.7	Carry Flag.
AC	PSW.6	Auxiliary Carry Flag.
F0	PSW.5	Flag 0 available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1 (SEE NOTE 1).
RS0	PSW.3	Register Bank selector bit 0 (SEE NOTE 1).
OV	PSW.2	Overflow Flag.
—	PSW.1	User definable flag.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of '1' bits in the accumulator.

NOTE:
1. The value presented by RS0 and RS1 selects the corresponding register bank.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

Figuur 7/6.2-26: Program Status Word (PSW), bit adresseerbaar.

SMOD	—	—	—	GF1	GF0	PD	IDL

SMOD	Double baud rate bit. If Timer 1 is used to generate baud rate and SMOD = 1, the baud rate is doubled when the Serial Port is used in modes 1, 2, or 3.
—	Not implemented, reserved for future use.*
—	Not implemented, reserved for future use.*
—	Not implemented, reserved for future use.*
GF1	General purpose flag bit.
GF0	General purpose flag bit.
PD	Power Down bit. Setting this bit activates Power Down operation in the 80C51BH. (Available only in CHMOS).
IDL	Idle Mode bit. Setting this bit activates Idle Mode operation in the 80C51BH. (Available only in CHMOS).

If 1s are written to PD and IDL at the same time, PD takes precedence.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

Figuur 7/6.2-27: Power Control Register (PCON), niet bit adresseerbaar.

Interrupt Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI & TI	0023H
TF2 & EXF2	002BH

Tabel 7/6.2-12: Interruptbronnen met de bijbehorende vectoradressen.

6.2 Algemene eigenschappen van de 8051-familie

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	—	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	Not implemented, reserved for future use.*
ET2	IE.5	Enable or disable the Timer 2 overflow or capture interrupt (8052 only).
ES	IE.4	Enable or disable the serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External Interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

Figuur 7/6.2-28: Interrupt Enable Register (IE), bit adresseerbaar.

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

—	—	PT2	PS	PT1	PX1	PT0	PX0

—	IP. 7	Not implemented, reserved for future use.*
—	IP. 6	Not implemented, reserved for future use.*
PT2	IP. 5	Defines the Timer 2 interrupt priority level (8052 only).
PS	IP. 4	Defines the Serial Port interrupt priority level.
PT1	IP. 3	Defines the Timer 1 interrupt priority level.
PX1	IP. 2	Defines External Interrupt 1 priority level.
PT0	IP. 1	Defines the Timer 0 interrupt priority level.
PX0	IP. 0	Defines the External Interrupt 0 priority level.

*User software should not write 1s to reserved bits. These bits may be used in future MCS-51 products to invoke new features. In that case, the reset or inactive value of the new bit will be 0, and its active value will be 1.

Figuur 7/6.2-29: Interrupt Priority Register (IP), bit adresseerbaar.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON. 7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON. 6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.
TF0	TCON. 5	Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.
TR0	TCON. 4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON. 3	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	TCON. 2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	TCON. 1	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.
IT0	TCON. 0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

Figuur 7/6.2-30: Timer/Counter Control Register (TCON), bit adresseerbaar.

6.2 Algemene eigenschappen van de 8051-familie

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
TIMER 1				TIMER 0			

GATE When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).

C/$\overline{T}$ Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).

M1 Mode selector bit. (NOTE 1)

M0 Mode selector bit. (NOTE 1)

NOTE 1:

M1	M0	Operating Mode	
0	0	0	13-bit Timer (MCS-48 compatible)
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

Figuur 7/6.2-31: Timer/Counter Mode Control Register (TMOD), niet bit adresseerbaar.

MODE	TIMER 0 FUNCTION	TMOD INTERNAL CONTROL (NOTE 1)	TMOD EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	08H
1	16-bit Timer	01H	09H
2	8-bit Auto-Reload	02H	0AH
3	two 8-bit Timers	03H	0BH

Tabel 7/6.2-13: Waarden van TMOD voor Timer/Counter 0, bij gebruik als timer.

MODE	TIMER 1 FUNCTION	TMOD INTERNAL CONTROL (NOTE 1)	TMOD EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	00H	80H
1	16-bit Timer	10H	90H
2	8-bit Auto-Reload	20H	A0H
3	does not run	30H	B0H

Tabel 7/6.2-15: Waarden van TMOD voor Timer/Counter 1, bij gebruik als timer.

MODE	COUNTER 0 FUNCTION	TMOD INTERNAL CONTROL (NOTE 1)	TMOD EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	04H	0CH
1	16-bit Timer	05H	0DH
2	8-bit Auto-Reload	06H	0EH
3	one 8-bit Counter	07H	0FH

NOTES:
1. The Timer is turned ON/OFF by setting/clearing bit TR0 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT0}$ when TR0 = 1 (hardware control).

Tabel 7/6.2-14: Waarden van TMOD voor Timer/Counter 0, bij gebruik als teller.

MODE	COUNTER 1 FUNCTION	TMOD INTERNAL CONTROL (NOTE 1)	TMOD EXTERNAL CONTROL (NOTE 2)
0	13-bit Timer	40H	C0H
1	16-bit Timer	50H	D0H
2	8-bit Auto-Reload	60H	E0H
3	not available	—	—

NOTES:
1. The Timer is turned ON/OFF by setting/clearing bit TR1 in the software.
2. The Timer is turned ON/OFF by the 1 to 0 transition on $\overline{INT1}$ when TR1 = 1 (hardware control).

Tabel 7/6.2-16: Waarden van TMOD voor Timer/Counter 1, bij gebruik als teller.

6.2 Algemene eigenschappen van de 8051-familie

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

TF2	T2CON. 7	Timer 2 overflow flag set by hardware and cleared by software. TF2 cannot be set when either RCLK = 1 or CLK = 1
EXF2	T2CON. 6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX, and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON. 5	Receive clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its receive clock in modes 1 & 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TLCK	T2CON. 4	Transmit clock flag. When set, causes the Serial Port to use Timer 2 overflow pulses for its transmit clock in modes 1 & 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON. 3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of negative transition on T2EX if Timer 2 is not being used to clock the Serial Port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON. 2	Software START/STOP control for Timer 2. A logic 1 starts the Timer.
C/$\overline{T2}$	T2CON. 1	Timer or Counter select. 0 = Internal Timer. 1 = External Event Counter (falling edge triggered).
CP/$\overline{RL2}$	T2CON. 0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, Auto-Reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the Timer is forced to Auto-Reload on Timer 2 overflow.

Figuur 7/6.2-32: Timer/Counter 2 Control Register (T2CON), bit adresseerbaar (alleen 8052).

MODE	T2CON	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	00H	08H
16-bit Capture	01H	09H
BAUD rate generator receive & transmit same baud rate	34H	36H
receive only	24H	26H
transmit only	14H	16H

Tabel 7/6.2-17: Timer/Counter 2 Set-Up, bij gebruik als timer. Behalve voor de baud-rate generatormode bevatten de waarden voor T2CON niet de instelling van het TR2-bit. Om de timer in te schakelen moet de TR2-bit daarom apart op 1 worden geset.

De volgende tabellen geven enkele waarden voor TMOD die gebruikt kunnen worden om Timer 0 voor verschillende modes klaar te zetten. Er wordt aangenomen dat slechts één timer tegelijk wordt gebruikt. Als het nodig is dat de timers 0 en 1 tegelijk werken, moet voor iedere mode de waarde in TMOD van timer 0 worden geORed met de waarde van timer 1 (tabellen 7/6.2-15 en -16).
Als het bijvoorbeeld gewenst is dat timer 0 in mode 1 GATE werkt (externe besturing) en timer 1 in mode 2 COUNTER, dan moet de waarde 69H in TMOD worden geladen (= 09H van tabel -13, geORed met 60H van tabel -16). Bovendien wordt aangenomen dat de gebruiker op dit punt de timers nog niet aanzet, maar dat op een later tijdstip in het programma doet door bit TRx (in TCON) op 1 te zetten.

MODE	TMOD	
	INTERNAL CONTROL (NOTE 1)	EXTERNAL CONTROL (NOTE 2)
16-bit Auto-Reload	02H	0AH
16-bit Capture	03H	0BH

NOTES:
1. Capture/Reload occurs only on Timer/Counter overflow.
2. Capture/Reload occurs on Timer/Counter overflow and a 1 to 0 transition on T2EX pin except when Timer 2 is used in the baud rate generating mode.

Tabel 7/6.2-18: Timer/Counter 2 Set-Up, bij gebruik als teller.

6.2 Algemene eigenschappen van de 8051-familie

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON. 7 Serial Port mode specifier. (NOTE 1).

SM1 SCON. 6 Serial Port mode specifier. (NOTE 1).

SM2 SCON. 5 Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 9).

REN SCON. 4 Set/Cleared by software to Enable/Disable reception.

TB8 SCON. 3 The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.

RB8 SCON. 2 In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.

TI SCON. 1 Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.

RI SCON. 0 Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

NOTE 1:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR Fosc./32
1	1	3	9-Bit UART	Variable

Figuur 7/6.2-33: Serial Port Control Register (SCON), bit adresseerbaar.

MODE	SCON	SM2 VARIATION
0 1 2 3	10H 50H 90H D0H	Single Processor Environment (SM2 = 0)
0 1 2 3	NA 70H B0H F0H	Multiprocessor Environment (SM2 = 1)

Tabel 7/6.2-19: Serial Port Set-Up.

Het genereren van Baud-rates

Seriële Poort in Mode 0

Mode 0 heeft een vaste baud-rate van 1/12 maal de oscillatorfrequentie. Om de seriële poort in deze mode te laten werken behoeft geen van de timers/tellers te worden ingesteld. Alleen het SCON-register moet worden gedefinieerd.

$$\text{Baud-rate} = \frac{\text{Osc. freq.}}{12}$$

Seriële Poort in Mode 1

Mode 1 heeft een variabele baud-rate die door Timer 1 of Timer 2 (alleen 8052) kan worden gegenereerd:

- Het opwekken van baud-rates met Timer/Counter 1:
  Voor dit doel wordt timer 1 in mode 2 gebruikt (Auto-Reload, zie timer 1 set-up tabel 7/6.2-15):

  $$\text{Baud-rate} = \frac{\text{K.Osc. freq.}}{32 \cdot 12 \cdot (256 - \text{TH1})}$$

  Als SMOD=0, dan K=1
  Als SMOD=1, dan K=2
  SMOD is het PCON-register.
  Meestal is de baud-rate bekend en behoeft men alleen de herlaadwaarde voor TH1 te weten.
  De formule om TH1 uit te rekenen is:

$$TH1 = 256 - \frac{K \cdot Osc.\ freq.}{384 \cdot baud\text{-}rate}$$

TH1 moet een geheel getal zijn. Door TH1 af te ronden op het dichtst bijzijnde gehele getal wordt niet altijd de gewenste baud-rate bereikt.
In dat geval kan het nodig zijn een andere kristal-frequentie te kiezen. Aangezien het PCON-register niet bit adresseerbaar is, is logisch ORen van het PCON-register een manier om toch de bit te setten (bijvoorbeeld ORL PCON,#80H). Het adres van PCON is 87H.

- Het opwekken van baud-rates met Timer/Counter 2:
Voor dit doel moet timer 2 in de baud-rate generating mode worden gebruikt (zie timer 2 set-up tabel 7/6.2-17). Als timer 2 via pen T2 wordt geklokt, is de baud-rate:

$$Baud\text{-}rate = \frac{Timer\ 2\ Overflow\text{-}rate}{16}$$

Als timer 2 intern wordt geklokt:

Baud-rate

$$= \frac{Osc.\ freq.}{32 \cdot [65536 - (RCAP2H,RCAP2L)]}$$

Om de herlaadwaarde voor RCAP2H en RCAP2L te verkrijgen kan bovenstaande vergelijking worden geschreven als:

$$RCAP2H,RCAP2L = \frac{Osc.\ freq.}{32 \cdot Baud\text{-}rate}$$

Seriële Poort in Mode 2
In deze mode is de baud-rate gefixeerd op 1/32 of 1/64 maal de oscillatorfrequentie, afhankelijk van de waarde van de SMOD-bit in het PCON-register. In deze mode wordt geen van de timers gebruikt en is de clock afkomstig van de interne fase2-clock.
SMOD=1, Baud-rate= 1/32 Osc. freq
SMOD=0, Baud-rate= 1/64 Osc. freq
De SMOD-bit wordt geset door: ORL PCON,#80H.

Seriële Poort in Mode 3
De baud-rate in mode 3 is variabel en wordt op precies dezelfde manier ingesteld als bij mode 1.

Instructieset

Tot slot is in de tabellen 7/6.2-20a tot en met -20d een samenvatting van de instructieset en in de tabellen 7/6.2-21a tot en met -21c een zeer handig overzicht van de opcodes (in hexadecimale volgorde) voor de 8051-familie te zien.

6.2 Algemene eigenschappen van de 8051-familie

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag C	Flag OV	Flag AC	Instruction	Flag C	Flag OV	Flag AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

- Rn — Register R7–R0 of the currently selected Register Bank.
- direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].
- @Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.
- #data — 8-bit constant included in instruction.
- #data 16 — 16-bit constant included in instruction.
- addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.
- addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.
- rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.
- bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

Tabel 7/6.2-20a: Overzicht van de instructieset, deel 1.

6.2 Algemene eigenschappen van de 8051-familie

8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12

Mnemonic		Description	Byte	Oscillator Period
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12

Tabel 7/6.2-20b: Overzicht van de instructieset, deel 2.

6.2 Algemene eigenschappen van de 8051-familie

8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Tabel 7/6.2-20c: Overzicht van de instructieset, deel 3.

6.2 Algemene eigenschappen van de 8051-familie

8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12

Tabel 7/6.2-20d: Overzicht van de instructieset, deel 4.

6.2 Algemene eigenschappen van de 8051-familie

Instruction Opcodes in Hexadecimal Order

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A, #data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr, code addr
31	2	ACALL	code addr
32	1	RETI	

Hex Code	Number of Bytes	Mnemonic	Operands
33	1	RLC	A
34	2	ADDC	A, #data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr, #data
44	2	ORL	A, #data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr, #data
54	2	ANL	A, #data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr, #data
64	2	XRL	A, #data
65	2	XRL	A,data addr

Tabel 7/6.2-21a: Overzicht van de opcodes, deel 1.

6.2 Algemene eigenschappen van de 8051-familie

Instruction Opcodes In Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	data addr, data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3

Tabel 7/6.2-21b: Overzicht van de opcodes, deel 2.

6.2 Algemene eigenschappen van de 8051-familie

Instruction Opcodes in Hexadecimal Order (Continued)

Hex Code	Number of Bytes	Mnemonic	Operands	Hex Code	Number of Bytes	Mnemonic	Operands
CC	1	XCH	A,R4	E6	1	MOV	A,@R0
CD	1	XCH	A,R5	E7	1	MOV	A,@R1
CE	1	XCH	A,R6	E8	1	MOV	A,R0
CF	1	XCH	A,R7	E9	1	MOV	A,R1
D0	2	POP	data addr	EA	1	MOV	A,R2
D1	2	ACALL	code addr	EB	1	MOV	A,R3
D2	2	SETB	bit addr	EC	1	MOV	A,R4
D3	1	SETB	C	ED	1	MOV	A,R5
D4	1	DA	A	EE	1	MOV	A,R6
D5	3	DJNZ	data addr,code addr	EF	1	MOV	A,R7
D6	1	XCHD	A,@R0	F0	1	MOVX	@DPTR,A
D7	1	XCHD	A,@R1	F1	2	ACALL	code addr
D8	2	DJNZ	R0,code addr	F2	1	MOVX	@R0,A
D9	2	DJNZ	R1,code addr	F3	1	MOVX	@R1,A
DA	2	DJNZ	R2,code addr	F4	1	CPL	A
DB	2	DJNZ	R3,code addr	F5	2	MOV	data addr,A
DC	2	DJNZ	R4,code addr	F6	1	MOV	@R0,A
DD	2	DJNZ	R5,code addr	F7	1	MOV	@R1,A
DE	2	DJNZ	R6,code addr	F8	1	MOV	R0,A
DF	2	DJNZ	R7,code addr	F9	1	MOV	R1,A
E0	1	MOVX	A,@DPTR	FA	1	MOV	R2,A
E1	2	AJMP	code addr	FB	1	MOV	R3,A
E2	1	MOVX	A,@R0	FC	1	MOV	R4,A
E3	1	MOVX	A,@R1	FD	1	MOV	R5,A
E4	1	CLR	A	FE	1	MOV	R6,A
E5	2	MOV	A,data addr	FF	1	MOV	R7,A

Tabel 7/6.2-21c: Overzicht van de opcodes, deel 3.

6.2 Algemene eigenschappen van de 8051-familie

7/6.3

De basistypen 8031, 8032, 8051, 8052, 8751 en 8752

Inleiding

NMOS-typen
In dit gedeelte worden de NMOS-typen van de 8051-familie microcontrollers behandeld. Door de geringe onderlinge verschillen kunnen de 8031, 8032, 8051, 8052, 8751 en 8752 tegelijk worden besproken.
Om vergissingen uit te sluiten worden de blokschema's, logische symbolen en behuizingen van de 8031/8051/8751 en van de 8032/8052/8752 apart getoond. Zoals in hoofdstuk 7/6.2 wordt uitgelegd staan voor byte- en numerieke operaties op kleine datastructuren enkele handige en snelle adresseringsmodes van de interne RAM ter beschikking. De instructieset omvat een ruime keus aan 8 bit rekenkundige instructies, waaronder vermenigvuldigen en delen, terwijl enkele bitmanipulaties en -testen mogelijk zijn.

Technologie
De hier behandelde AH- en H-typen worden door Intel met het HMOS II-proces vervaardigd (de 8751H-8 met een HMOS-E proces). Bij andere fabrikanten worden andere processen toegepast, zoals MYMOS-III bij Siemens. In tabel 7/6.3-1 worden de onderlinge verschillen (en overeenkomsten) van deze groep microcontrollers verduidelijkt.
De 8031AH en 8032AH bevatten geen ROM maar wel 128 bytes RAM. De 8051AH heeft 4 kB ROM en 128 bytes RAM aan boord; de 8052AH is tweemaal zo groot (zowel ROM als RAM) en "achter"waarts compatibel met de 8051AH. De 8751H is de EPROM-versie van de 8051AH (elektrisch programmeerbaar en UV-wisbaar) en is daarmee dan ook volledig compatibel. De 8071H heeft echter een extra eigenschap: een Program Memory Security bit, waarmee onbevoegd uitlezen van het programma kan worden voorkomen. De 8051AHP heeft deze zelfde eigenschap. De 8751H-8 is identiek aan de 8751H, maar werkt op 8 MHz. De 8751BH is een OTP-uitvoering (One Time Programmable) van de 8751H: hij heeft geen kwartsvenstertje en kan dus ook niet gewist worden. De 8752BH is tenslotte de EPROM-versie van de 8052.

Kenmerken
- 8-bit microcontrollers
- Boole'se processor
- 111 instructies (64 één cyclus)
- 32 in-/uitgangslijnen (vier 8 bit I/O-poorten)
- interrupt-prioriteit op 2 niveaus
- bit adresseerbare RAM
- 64 kB programmageheugenruimte
- 64 kB datageheugenruimte
- programmeerbaar full-duplex serieel kanaal
- enkele +5 V voeding (programmeerspanning EPROM-type: +21 V)
- behuizingen: 40-pens plastic of ceramische DIP of 44-pens PLCC (figuur 7/6.3-1 en -4), (8751 alleen 40-pens CERDIP)
- bedrijfstemperatuur: 0 - 70 C
- fabrikanten o.a.:
  Intel: 8031AH, 8032AH, 8051AH, 8051AHP, 8052AH, 8751H(-8), 8751BH, 8752BH

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-8	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

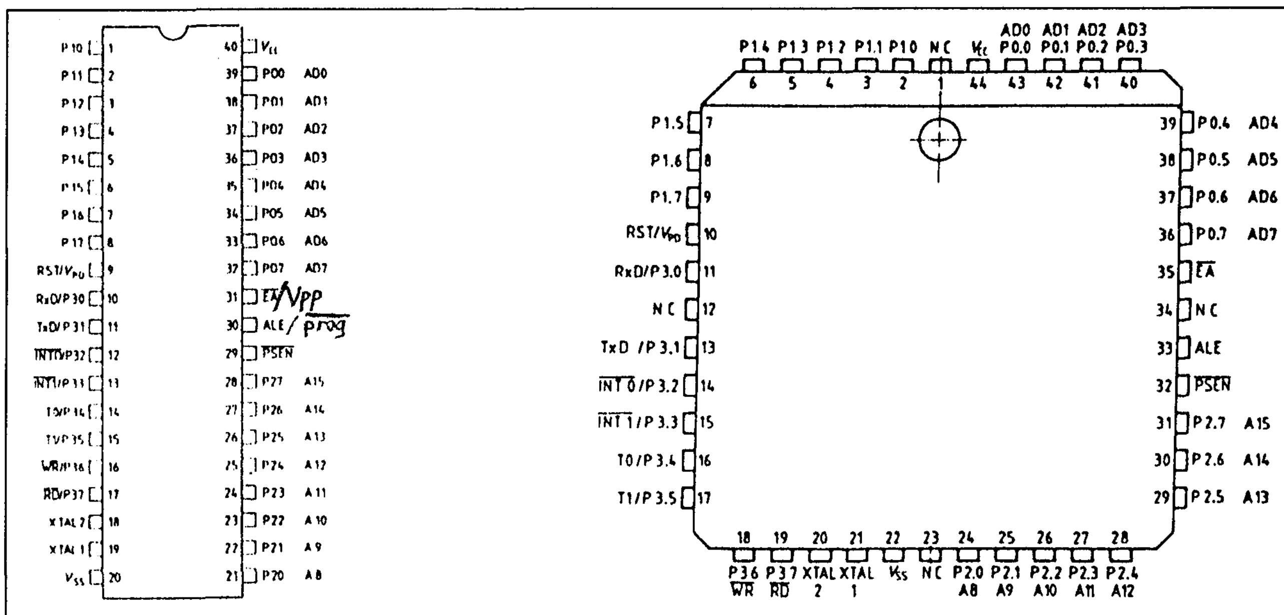
Tabel 7/6.3-1: Overzicht van verschillen en overeenkomsten van de MCS-51 microcontrollers.

Siemens: SAB8031A(-16), SAB8032A (-16), SAB8051A (-16), SAB8052A
Philips: MAB8031AH, MAB8032AH, MAB8032AH, MAB8052AH

Aansluitingen en penfuncties

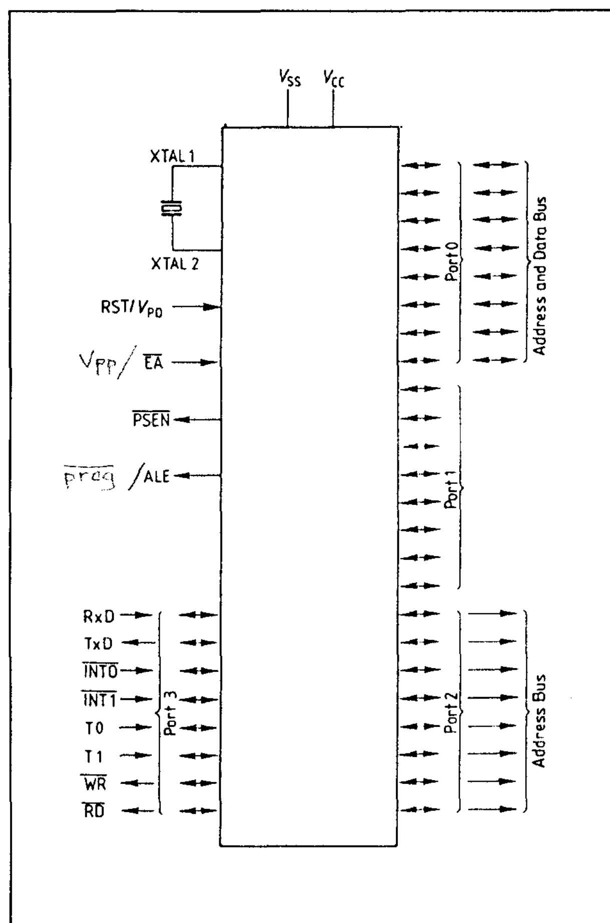
Behuizingen

De hier behandelde microcontrollers zijn leverbaar in twee soorten behuizingen: de 40-pens DIL en de 44-pens PLCC behuizing. In figuur 7/6.3-1 is voor de 8031, 8051 en 8751 van beide behuizingen het bovenaanzicht getekend. De 8751 is alleen in de 40-pens ceramische uitvoering leverbaar, waarbij een kwartsvenstertje het wissen met ultraviolet licht mogelijk maakt. Figuur 7/6.3-2 geeft het bijbehorende logische symbool voor deze typen en figuur 7/6.3-3 het volledige blokschema. De 8032, 8052 en 8752 hebben drie tellers in plaats van twee en hebben dus een extra signaal- en triggeringang nodig. Vandaar dat de aansluitingen hiervan (T2 en TX2) in de figuren 7/6.3-4 en -5 zijn opgenomen. De hieronder gebruikte pennummering heeft voor alle typen betrekking op de DIL-behuizing. PROG en V_{PP} hebben natuurlijk alleen betrekking op de EPROM/OTP-typen 8751 en 8752.



Figuur 7/6.3-1: Aansluitingen van de 8031/8051/8751 (40-pens DIL en 44-pens PLCC).

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



Figuur 7/6.3-2: Logisch symbool van de 8031/8051/8751.

Port Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 External Input)
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger)

Tabel 7/6.3-2: Alternatieve functies van de poort 1-pennen 0 en 1.

Poort 1 (I/O), pennen 1 - 8

Poort 1 is een 8 bit quasi bidirectionele in-/uitgangspoort met interne optrekweerstanden. De uitgangsbuffers kunnen 4 LSTTL-belastingen source/sinken. Poort 1 wordt tijdens het programmeren van de EPROM (8751 of 8752) en tijdens het verifiëren van de ROM/EPROM gebruikt voor de lage adresbytes.
Tevens bevat poort 1 (alleen bij de 8032, 8052 en 8752) de aansluitpennen voor timer/counter 2 (zie tabel 7/6.3-2).

RST/Vpd (I), pen 9

De microcontroller wordt door een logisch HOOG signaal op deze ingang gereset. Alleen bij de Siemens typen levert deze ingang de standby-voeding voor de RAM als V_{CC} te laag wordt (power down).

Poort 3 (I/O), pennen 10 - 17

Poort 3 is een quasi bidirectionele in-/uitgangspoort, waarvan de uitgangsbuffer 4 LSTTL-belastingen kan sinken/sourcen. Poort 3 bevat tevens de interrupt, timer, seriële poort en $\overline{RD}$- en $\overline{WR}$-pennen. De uitgangslatch voor een secundaire functie moet op 1 worden geprogrammeerd om die functie uitvoerbaar te maken (tabel 7/6.3-3).

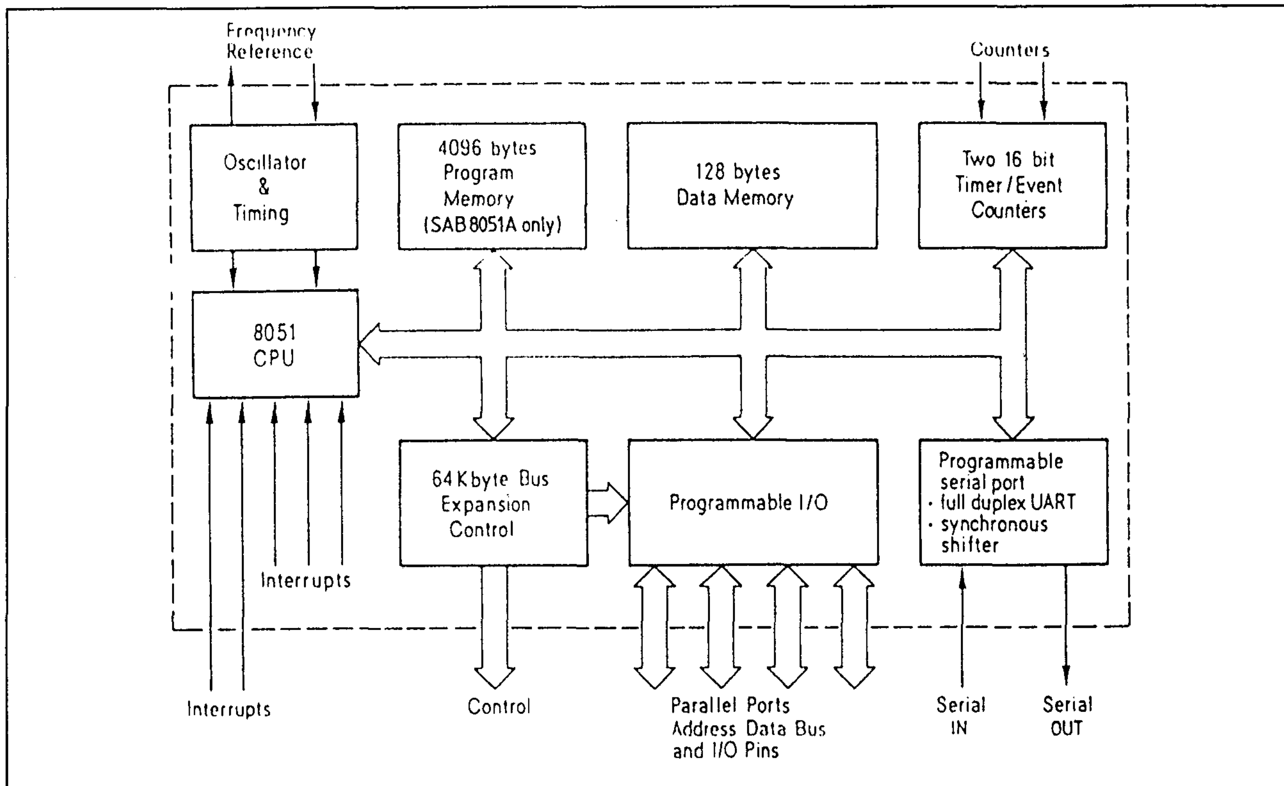
Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

Tabel 7/6.3-3: Alternatieve functies van de poort 3-pennen.

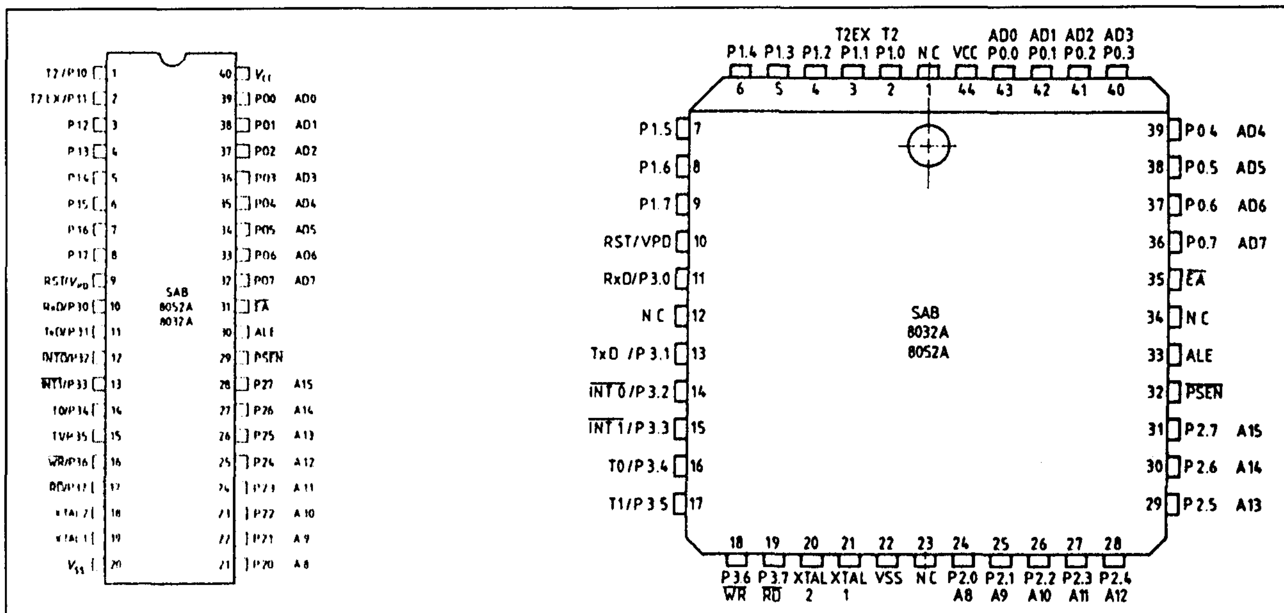
XTAL1, XTAL2, pennen 18 en 19

XTAL1 is de ingang van de oscillatorversterker waarop een kristal kan worden aangesloten (aan V_{SS} leggen als een externe oscillator wordt gebruikt). XTAL2 is de uitgang van de oscillatorversterker voor aansluiting van het kristal of een externe oscillator (figuur 7/6.3-7).

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752

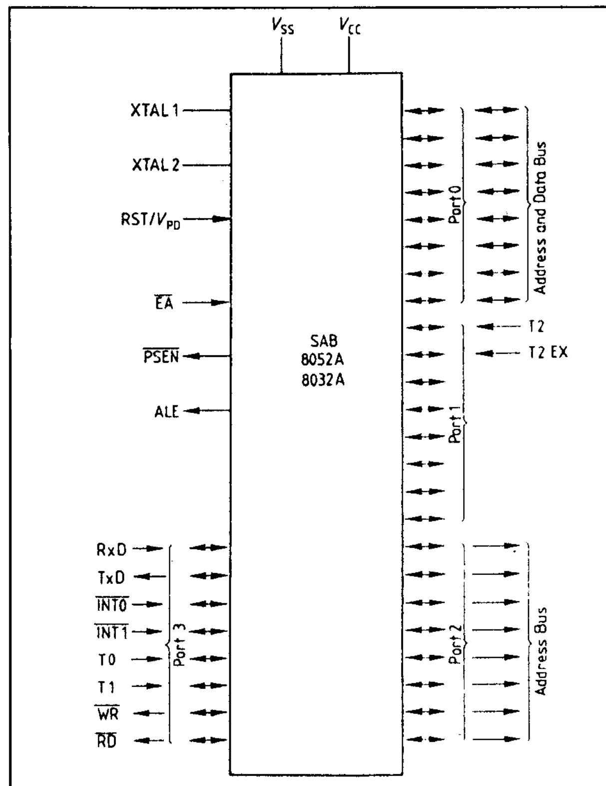


Figuur 7/6.3-3: Volledig blokschema van de 8031/8051/8751.

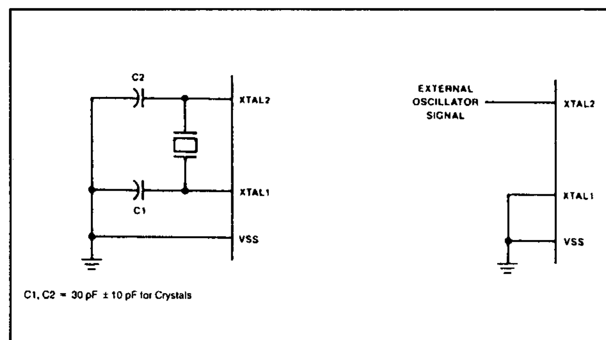


Figuur 7/6.3-4: Aansluitingen van de 8032/8052/8752 (bovenaanzicht).

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



Figuur 7/6.3-5: Logisch sym1bool van de 8032/8052/8752.



Figuur 7/6.3-7: Aansluitingen van de oscillator (links met kristal, rechts met externe frequentiebron).

V_{SS}, pen 20
Circuit-aarde.

Poort 2 (I/O), pennen 21 - 28
Poort 2 is een quasi bidirectionele in-/uitgangspoort, waarvan de uitgangs buffer 4 LSTTL-belastingen kan sinken/sourcen. Ook komen op poort 2 bij het adresseren van 16 bit extern geheugen de hoge adresbytes te staan. Bovendien ontvangt poort 2 de hoge adresbytes tijdens het programmeren van EPROM (8751 of 8752) en het verifiëren van ROM/EPROM.

$\overline{\text{PSEN}}$ (O), pen 29
Het Program Store Enable-signaal wordt gebruikt om het externe programmageheugen op de bus te zetten. Het wordt elke zesde oscillatorperiode geaktiveerd, behalve bij externe datageheugen toegangen.

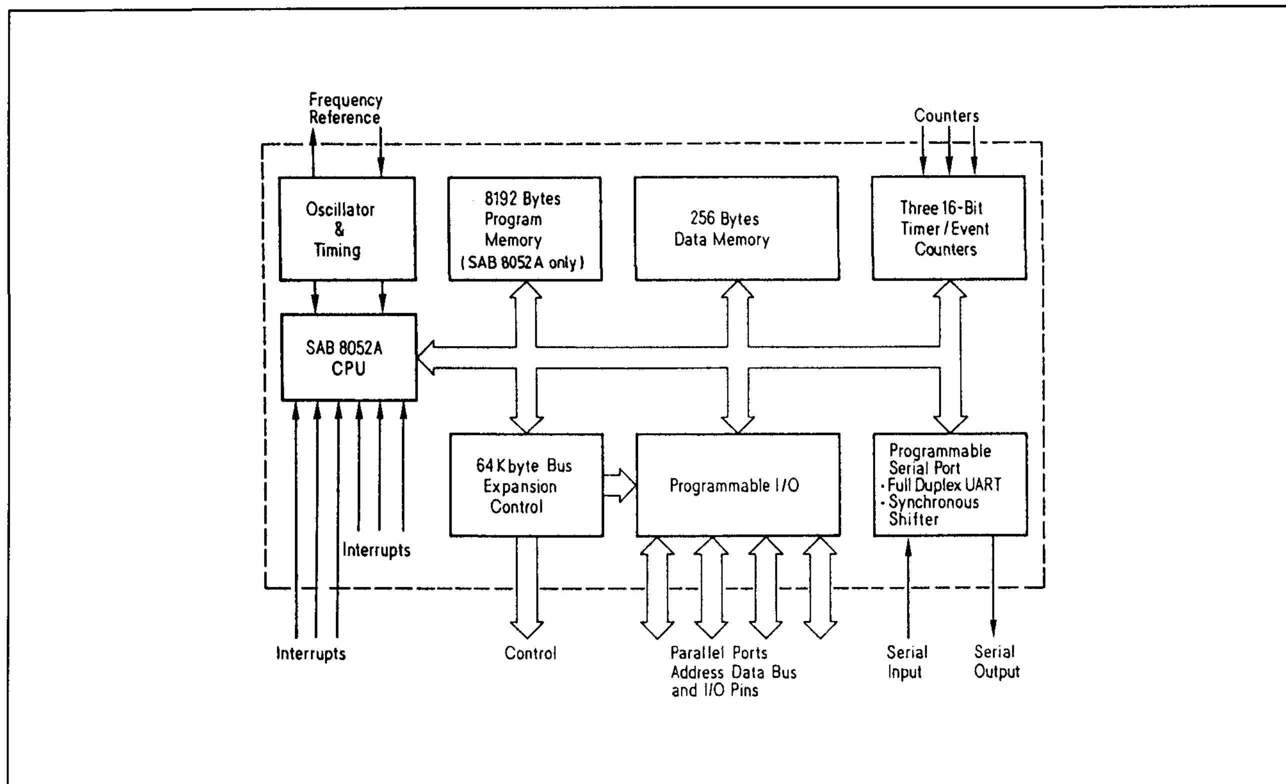
ALE/$\overline{\text{PROG}}$ (O), pen 30
Levert het latch enablesignaal om bij normaal bedrijf de lage byte van een adres in het externe geheugen te latchen. Het signaal wordt elke zesde oscillatorperiode aktief, behalve bij externe datageheugen toegangen. Bij de 8751 en 8752 wordt deze pen gebruikt voor ontvangst van de programmeerpuls voor de EPROM.

$\overline{\text{EA}}$/Vpp (I), pen 31
Wanneer deze ingang HOOG is, worden alleen instructies uit intern geheugen uitgevoerd; door de External Access ingang LAAG te maken wordt code opgehaald uit externe programma-adressen tussen 0000H tot en met FFFFH. Bij de 8031 moet deze ingang altijd LAAG zijn. Bij de 8751 en 8752 komt op deze pen de 21 V programmeerspanning voor de EPROM te staan.

Poort 0 (I/O), pennen 32 - 39
Poort 0 is een 8 bit open-drain bidirectionele in-/uitgangspoort. Elke pen kan een sinkstroom voor 8 LSTTL-belastingen leveren. Bij gebruik van extern geheugen dient poort 0 tevens als het gemultiplexte lage adresbyte en de databus. Bij het programmeren van de EPROM in de 8751 of 8752 ontvangt poort 0 de codebytes en brengt die naar buiten tijdens verificatie van de ROM/EPROM.

V_{CC}, pen 40
Voedingsspanning.

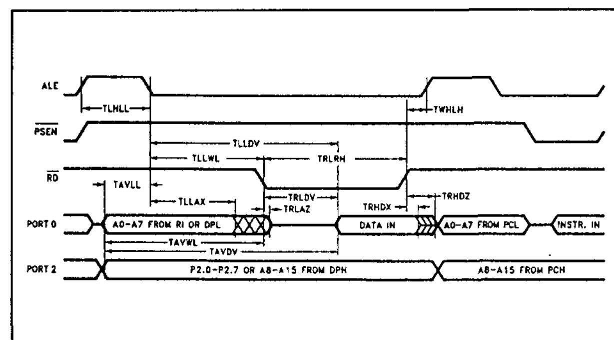
6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



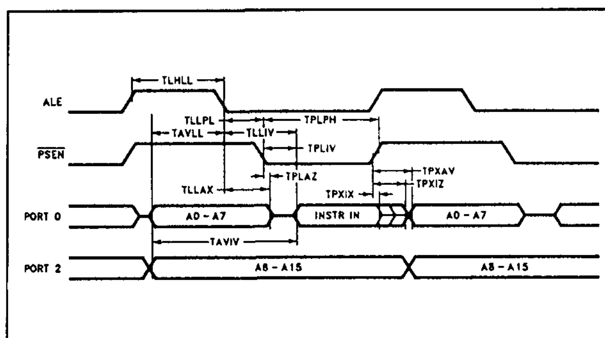
Figuur 7/6.3-6: Volledig blokschema van de 8032/8052/8752.

Overige algemene kenmerken

De instructieset voor de hier behandelde microcontrollers is reeds opgenomen in hoofdstuk 7/6.2. De elektrische en timingeigenschappen van de Intel-typen zijn te zien in de tabel 7/6.3-4 tot en met figuur 7/6.3-12.



Figuur 7/6.3-9: Uitlezen van extern datageheugen (zie ook tabel 7/6.3-6).



Figuur 7/6.3-8: Uitlezen van extern programmageheugen (tabel 7/6.3-6).

Ambient Temperature Under Bias	− 40°C to + 85°C
Storage Temperature	− 65°C to + 150°C
Voltage on $\overline{EA}/V_{PP}$ Pin to V_{SS}	− 0.5V to + 21.5V
Voltage on Any Other Pin to V_{SS}	− 0.5V to + 7V
Power Dissipation	1.5W

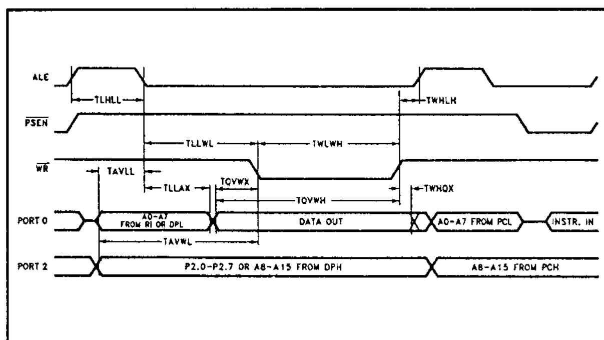
Tabel 7/6.3-4: De maximaal toegelaten waarden (voor de Intel-typen).

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752

DC CHARACTERISTICS T_A (Under Bias) = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter		Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$ Pin of 8751H & 8751H-8)		−0.5	0.8	V	
V_{IL1}	Input Low Voltage to $\overline{EA}$ Pin of 8751H & 8751H-8		0	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)		2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage to XTAL2, RST		2.5	V_{CC} + 0.5	V	XTAL1 = V_{SS}
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*			0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)*					
		8751H, 8751H-8		0.60 0.45	V V	I_{OL} = 3.2 mA I_{OL} = 2.4 mA
		All Others		0.45	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)		2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)		2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, RST) 8032AH, 8052AH All Others			−800 −500	μA μA	V_{IN} = 0.45V V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8 Only			−15	mA	V_{IN} = 0.45V
I_{IL2}	Logical 0 Input Current (XTAL2)			−3.2	mA	V_{IN} = 0.45V
I_{LI}	Input Leakage Current (Port 0) 8751H & 8751H-8 All Others			±100 ±10	μA μA	0.45 ≤ V_{IN} ≤ V_{CC} 0.45 ≤ V_{IN} ≤ V_{CC}
I_{IH}	Logical 1 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8			500	μA	V_{IN} = 2.4V
I_{IH1}	Input Current to RST to Activate Reset			500	μA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current: 8031AH/8051AH 8032AH/8052AH 8751H/8751H-8			125 175 250	mA mA mA	All Outputs Disconnected; $\overline{EA}$ = V_{CC}
C_{IO}	Pin Capacitance			10	pF	Test freq = 1 MHz

Tabel 7/6.3-5: Gelijkspanningswaarden.



Figuur 7/6.3-10: Schrijven naar extern datageheugen (zie ook tabel 7/6.3-6).

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL − 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL − 117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL − 133	ns

Tabel 7/6.3-7: De timing van de seriële poort in schuifregister mode (zie ook figuur 7/6.3-11).

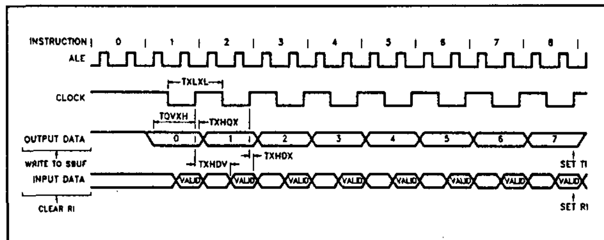
6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL − 40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL − 40		ns
TLLAX	Address Hold after ALE Low	48		TCLCL − 35		ns
TLLIV	ALE Low to Valid Instr In 8751H All Others	 	 183 233		 4TCLCL − 150 4TCLCL − 100	 ns ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL − 25		ns
TPLPH	$\overline{PSEN}$ Pulse Width 8751H All Others	 190 215		 3TCLCL − 60 3TCLCL − 35		 ns ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In 8751H All Others		 100 125		 3TCLCL − 150 3TCLCL − 125	 ns ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		63		TCLCL − 20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL − 8		ns
TAVIV	Address to Valid Instr In 8751H All Others		 267 302		 5TCLCL − 150 5TCLCL − 115	 ns ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL − 100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL − 100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL − 165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		97		2TCLCL − 70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL − 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL − 165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL − 50	3TCLCL + 50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL − 130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition 8751H All Others	 13 23		 TCLCL − 70 TCLCL − 60		 ns ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL − 150		ns
TWHQX	Data Hold after $\overline{WR}$	33		TCLCL − 50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High 8751H All Others	 33 43	 133 123	 TCLCL − 50 TCLCL − 40	 TCLCL + 50 TCLCL + 40	 ns ns

NOTE:
*This table does not include the 8751-8 AC characteristics

Tabel 7/6.3-6: Timing en schakeltijden (voor alle typen behalve de 8751H-8).

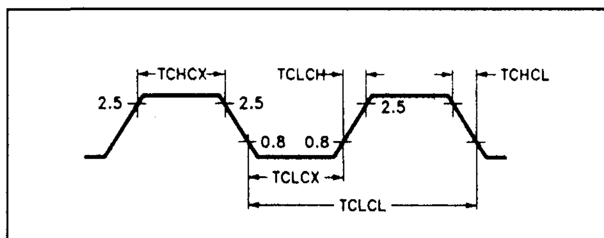
6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



Figuur 7/6.3-11: Golfvormen en schakeltijden van het schuifregister (zie ook tabel 7/6.3-7).

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency (except 8751H-8) 8751H-8	3.5 3.5	12 8	MHz MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

Tabel 7/6.3-8: Eisen, waaraan de externe clock moet voldoen.



Figuur 7/6.3-12: Golfvormen van de externe clock (zie tabel 7/6.3-8).

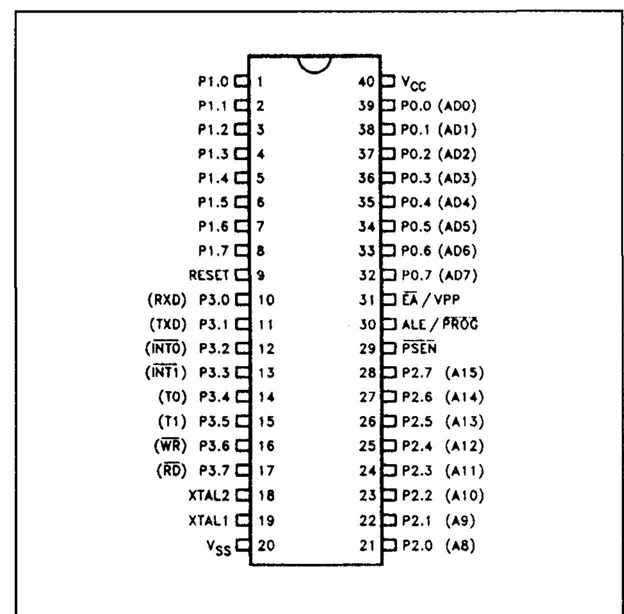
Programmeren van de EPROM's

Inleiding

Aangezien de 8751H, 8751BH en 8752BH onderling wat verschillend zijn, worden ze apart behandeld. Waar ze overeenkomsten hebben, wordt naar een ander type verwezen. Om de microcontrollers met EPROM te programmeren moet de clockfrequentie 4 tot 6 MHz zijn. Dit is nodig omdat de interne bus wordt gebruikt voor het overbrengen van adressen en data naar de betreffende interne registers.

Programmeren van de 8751H

De 8751H heeft een 4 kB EPROM die met UV-licht gewist kan worden en bevindt zich in een ceramische DIL-behuizing (figuur 7/6.3-13). Het adres van de EPROM-lokatie die geprogrammeerd moet worden, wordt op poort 1 en de pennen P2.0 tot en met P2.3 van poort 2 gezet, terwijl het codebyte die daar moet komen, op poort 0 wordt gezet. De overige pennen van poort 2 plus RST, $\overline{\text{PSEN}}$ en $\overline{\text{EA}}/V_{PP}$ moeten op de "programmeer"-niveaus van tabel 7/6.3-9 worden gehouden. Op ALE wordt gedurende 50 ms LAAG gehouden om het codebyte te programmeren. In figuur 7/6.3-14 is de programmeeropstelling te zien. Meestal wordt $\overline{\text{EA}}$ HOOG gehouden tot vlak voordat ALE een puls ontvangt. Dan wordt $\overline{\text{EA}}$ op +21 V gebracht, ontvangt EA een puls en gaat $\overline{\text{EA}}$ weer terug naar logisch HOOG (zie figuur 7/6.3-15).



Figuur 7/6.3-13: Aansluitingen van de 8751H.

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752

Mode	RST	PSEN	ALE	EA	P2.7	P2.6	P2.5	P2.4
Program	1	0	0*	VPP	1	0	X	X
Inhibit	1	0	1	X	1	0	X	X
Verify	1	0	1	1	0	0	X	X
Security Set	1	0	0*	VPP	1	1	X	X

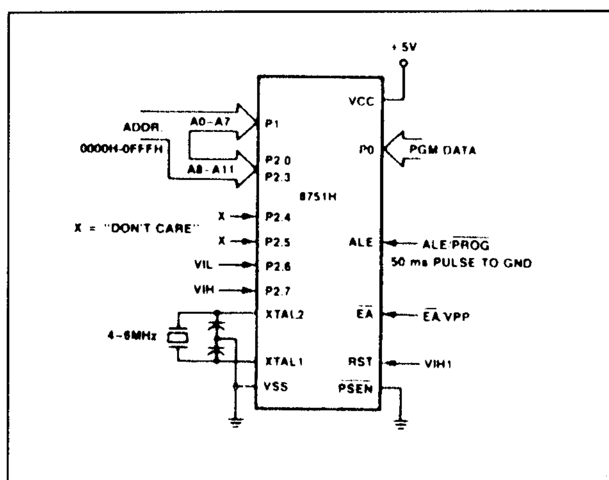
NOTE:
"1" = logic high for that pin
"0" = logic low for that pin
"X" = "don't care"
"VPP" = +21V ±0.5V
*ALE is pulsed low for 50 ms.

Tabel 7/6.3-9: EPROM programmeermodes voor de 8751H.

T_A = 21°C to 27°C; VCC = 5V ±10%; VSS = 0V

Symbol	Parameter	Min	Max	Units
VPP	Programming Supply Voltage	20.5	21.5	V
IPP	Programming Supply Current		30	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to PROG Low	48TCLCL		
TGHAX	Address Hold after PROG	48TCLCL		
TDVGL	Data Setup to PROG Low	48TCLCL		
TGHDX	Data Hold after PROG	48TCLCL		
TEHSH	P2.7 (ENABLE) High to VPP	48TCLCL		
TSHGL	VPP Setup to PROG Low	10		μs
TGHSL	VPP Hold after PROG	10		μs
TGLGH	PROG Width	45	55	ms
TAVQV	Address to Data Valid		48TCLCL	
TELQV	ENABLE Low to Data Valid		48TCLCL	
TEHQZ	Data Float after ENABLE	0	48TCLCL	

Tabel 7/6.3-10: Eisen voor programmeren en verifiëren van de 8751H.



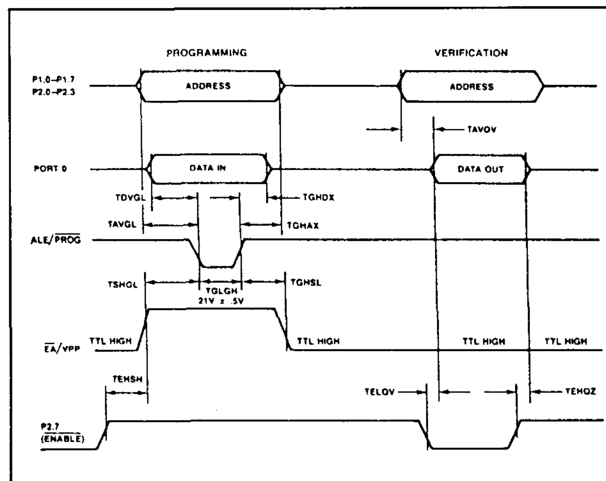
Figuur 7/6.3-14: Configuratie voor het programmeren van de 8751H.

Programmaverificatie

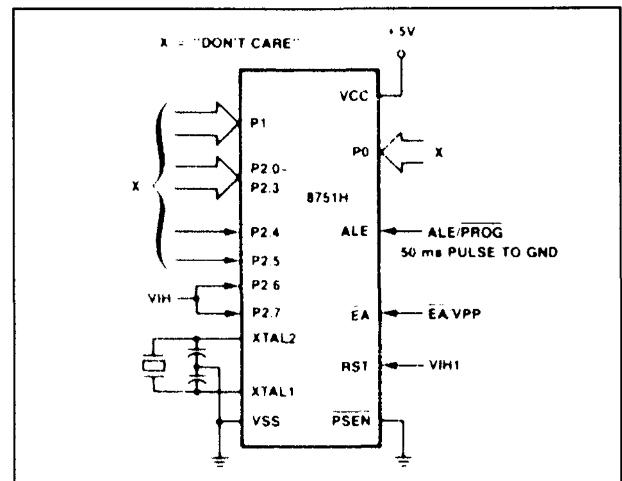
Als de beveiligingsbit (Security bit) niet geprogrammeerd is, kan het programmageheugen op de chip ter verificatie worden uitgelezen.

Dit kan tijdens en na het programmeren. Het adres van de lokatie in het programmageheugen die moet worden gelezen, wordt op poort 1 en de pennen P2.0 tot en met P2.3 gezet. De andere pennen moeten op de "verify"-niveaus uit tabel 7/6.3-9 worden gehouden. De inhoud van de geadresseerde lokatie verschijnt nu op poort 0 (die voor dit doel externe optrekweerstanden nodig heeft). Een en ander is te zien in figuur 7/6.3-16.

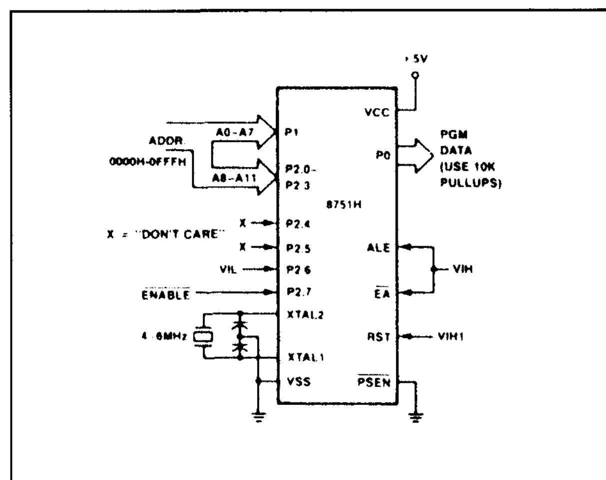
6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



Figuur 7/6.3-15: Timing en golfvormen bij programmeren en verifiëren van de EPROM in de 8751H.



Figuur 7/6.3-17: Het programmeren van de beveiligingsbit (Security-bit).



Figuur 7/6.3-16: Opstelling voor het verifiëren van de 8751H.

8751H EPROM-beveiliging

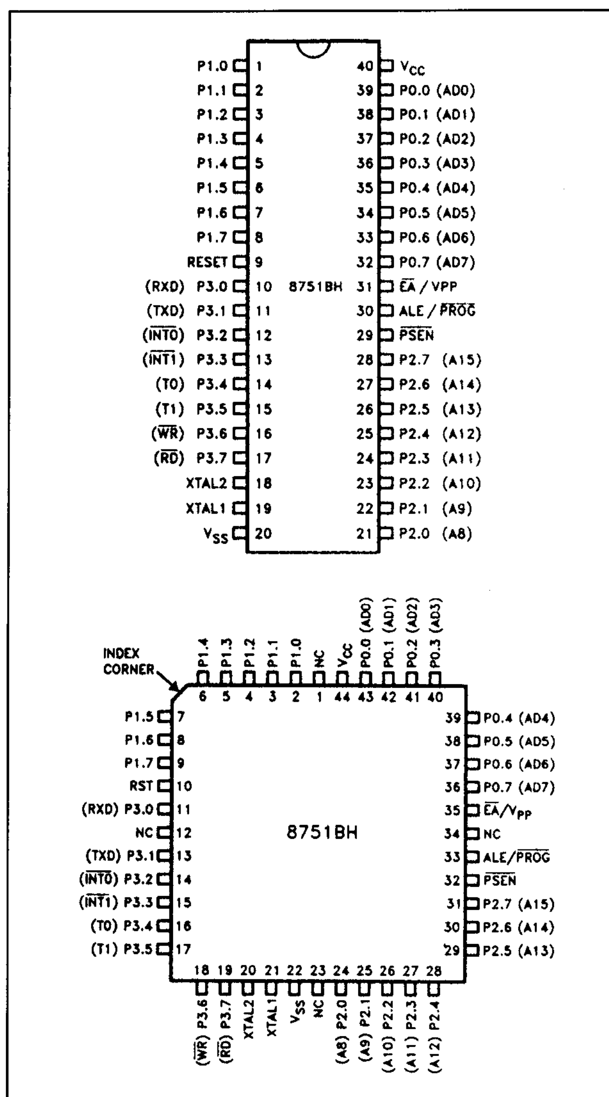
De beveiliging bestaat uit een "locking" bit. Wanneer dit geprogrammeerd is, is geen toegang van buitenaf tot het programmageheugen op de chip mogelijk. Figuur 7/6.3-17 toont het programmeren ervan: pen P2.6 wordt bij het programmeren HOOG gehouden. De toestand op de pennen van poort 0, poort 1 en P2.0 tot en met P2.3 doet er niet toe, terwijl de overige op de "security"-niveaus van tabel 7/6.3-9 moeten staan.

Nadat het beveiligingsbit geprogrammeerd is, kan dit slechts gecleared worden door het gehele programmageheugen te wissen. In geprogrammeerde toestand kan het interne programmageheugen niet worden uitgelezen, kan de controller niet verder worden geprogrammeerd en kan niet vanuit extern programmageheugen worden gewerkt.
Door de EPROM (inclusief beveiligingsbit) te wissen komt de volledige functionaliteit terug.

Wissen van de 8751H

De EPROM wordt gewist door hem bloot te stellen aan licht met een golflengte van korter dan 400 nm. Aangezien zonlicht en fluorescerende verlichting (TL) golflengten in dit gebied hebben, kan blootstelling hieraan (1 week zonlicht of 3 jaar TL) onbedoeld wissen veroorzaken. Het wordt daarom aanbevolen het venstertje te bedekken met een niet doorzichtig stickertje. De aanbevolen wisprocedure is bestraling met UV-licht (253,7 nm) gedurende een geïntegreerde dosis van minstens 15 Wsec/cm^2. Dit wordt bereikt met een ultraviolette lamp van 12 mW/cm^2 gedurende 20 à 30 minuten. Na het wissen bevindt de gehele EPROM zich in de "1" toestand.

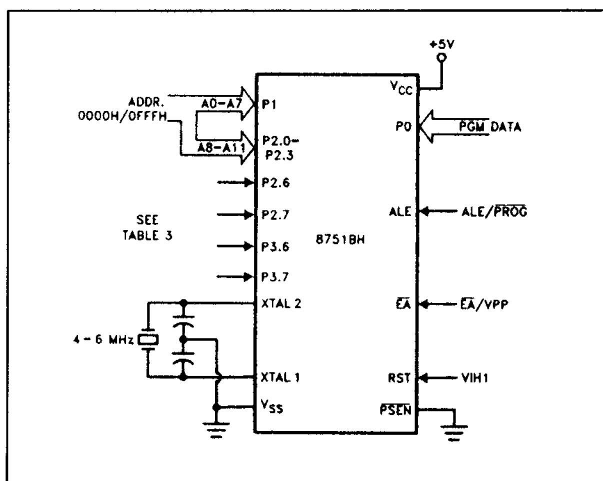
6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



Figuur 7/6.3-18: De 8751BH is opgenomen in een plastic behuizing zonder kwartsvenster.

Programmeren van de 8751BH

De 8751BH heeft net als de 8751H een 4 kB EPROM aan boord. Doordat de 8751BH echter in een plastic behuizing zonder venstertje is opgenomen (figuur 7/6.3-18) kan deze NIET gewist worden. Het programmeren van de 8751BH gaat op dezelfde manier als hiervoor beschreven bij de 8751H, behalve dat ALE/$\overline{\text{PROG}}$ nu niet 50 ms LAAG wordt gehouden, maar de vorm van een pulstrein heeft. Let ook op dat de programmeerspanning op $\overline{\text{EA}}$/V_{PP} nu 12,75 V bedraagt! Omdat nu P3.6 en P3.7 ook meedoen gelden de programmeeropstelling van figuur 7/6.3-19 en tabel 7/6.3-11.



Figuur 7/6.3-19: Opstelling voor het programmeren van de 8751BH.

Quick-pulse programmeer-algorithme

De 8751BH kan worden geprogrammeerd met de Quick-Pulse Programming Algorithm voor microcontrollers. Deze wordt gekenmerkt door een lagere programmeerspanning (12,75 V) en een kortere programmeerpuls (pulstrein). Het is hiermee mogelijk de gehele EPROM binnen 13 seconden te programmeren.

Wanneer V_{PP} op 12,75 V +/-0,25 V is gebracht, moet ALE/$\overline{\text{PROG}}$ 25 maal gedurende 100 µs LAAG worden gepulst. Hierna moet de zojuist geprogrammeerde byte worden geverifieerd, terwijl na afloop verificatie van de gehele array nodig is (figuur 7/6.3-20).

Verificatie van Programma en Signature

Als de beveiligingsbits (lockbits) niet geprogrammeerd zijn, kan het programmageheugen op de chip worden uitgelezen (zowel tijdens als na het programmeren). Het adres van de lokatie in het programmageheugen dat moet worden gelezen, wordt op poort 1 en de pennen P2.0 tot en met P2.3 gezet.

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752

MODE	RST	$\overline{PSEN}$	ALE/ $\overline{PROG}$	$\overline{EA}$/ V_{PP}	P2.7	P2.6	P3.6	P3.7
Program Code Data	1	0	0*	V_{PP}	1	0	1	1
Verify Code Data	1	0	1	1	0	0	1	1
Program Encryption Table Use Addresses 0-1FH	1	0	0*	V_{PP}	1	0	0	1
Program Lock Bits (LBx) x = 1	1	0	0*	V_{PP}	1	1	1	1
Program Lock Bits (LBx) x = 2	1	0	0*	V_{PP}	1	1	0	0
Read Signature	1	0	1	1	0	0	0	0

NOTES:
"1" = Valid high for that pin
"0" = Valid low for that pin
"V_{PP}" = +12.75V ±0.25V
* ALE/$\overline{PROG}$ is pulsed low for 100 μs for programming. (Quick-Pulse Programming)

Tabel 7/6.3-11: Programmeermodes voor de 8751BH en 8752BH.

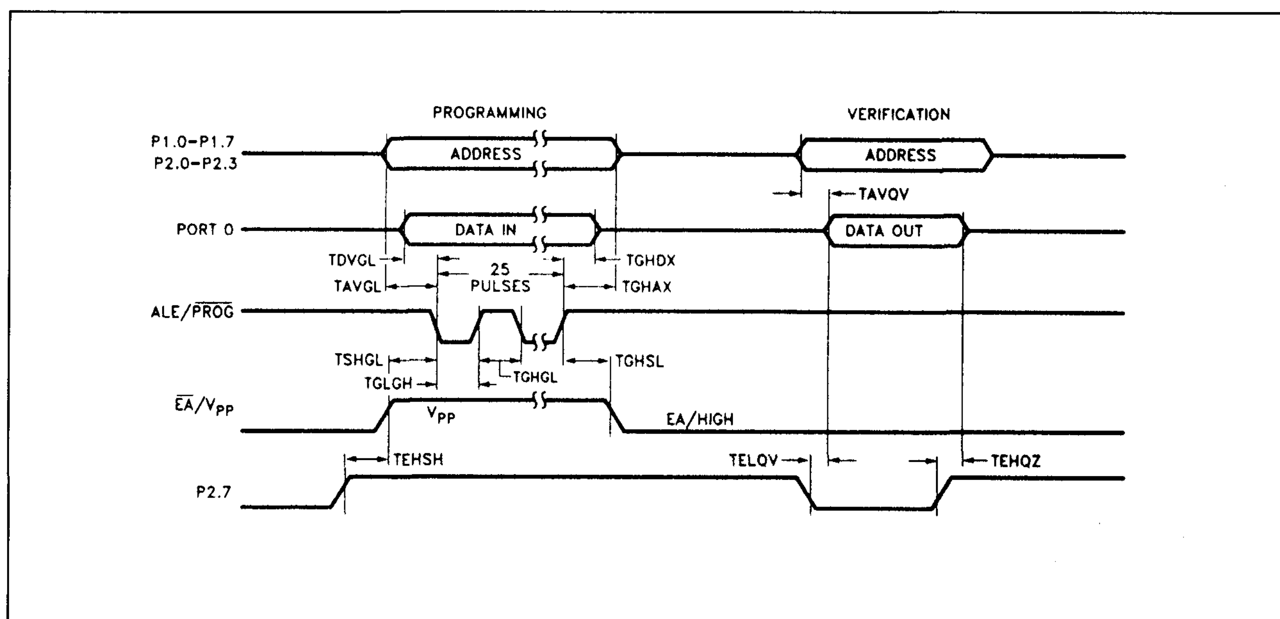
(T_A = 21°C to 27°C, V_{CC} = 5.0V ±10%, V_{SS} = 0V)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
IPP	Programming Supply Current		50	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$ Low	48TCLCL		
TGHAX	Address Hold After $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$ Low	48TCLCL		
TGHDX	Data Hold After $\overline{PROG}$	48TCLCL		
TEHSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		μsec
TGHSL	V_{PP} Hold After $\overline{PROG}$	10		μsec
TGLGH	$\overline{PROG}$ Width	90	110	μsec
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{ENABLE}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float After $\overline{ENABLE}$	0	48TCLCL	
TGHGL	$\overline{PROG}$ High to $\overline{PROG}$ Low	10		μsec

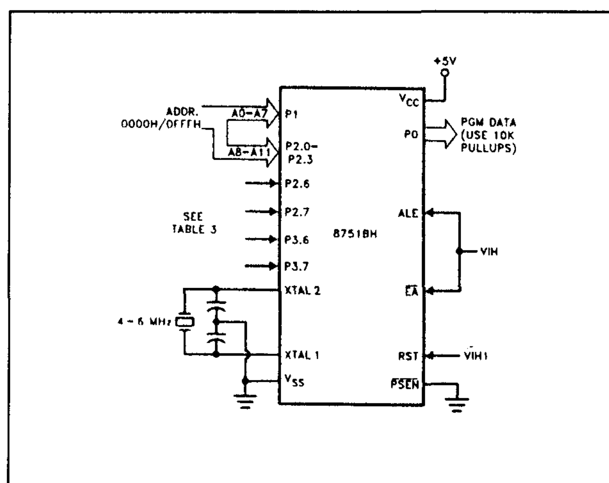
Tabel 7/6.3-12: Elektrische en timing karakteristieken van het programmeren en verifiëren van de 8751BH en 8752BH.

De andere pennen moeten op "verify"-niveaus uit tabel 7/6.3-11 hebben (figuur 7/6.3-21).
De inhoud van de gekozen lokatie verschijnt op poort 0, die externe optrekweerstanden moet hebben. Als de Encryptie Array in de EPROM werd geprogrammeerd, is de data op poort 0 de XNOR-functie van de codedata en de encryptiedata.
De gebruiker moet dus de encryptie-array kennen om "met de hand" de data te kunnen verifiëren.

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



Figuur 7/6.3-20: Timing en golfvormen bij het programmeren en verifiëren van de 8751BH en 8752BH.



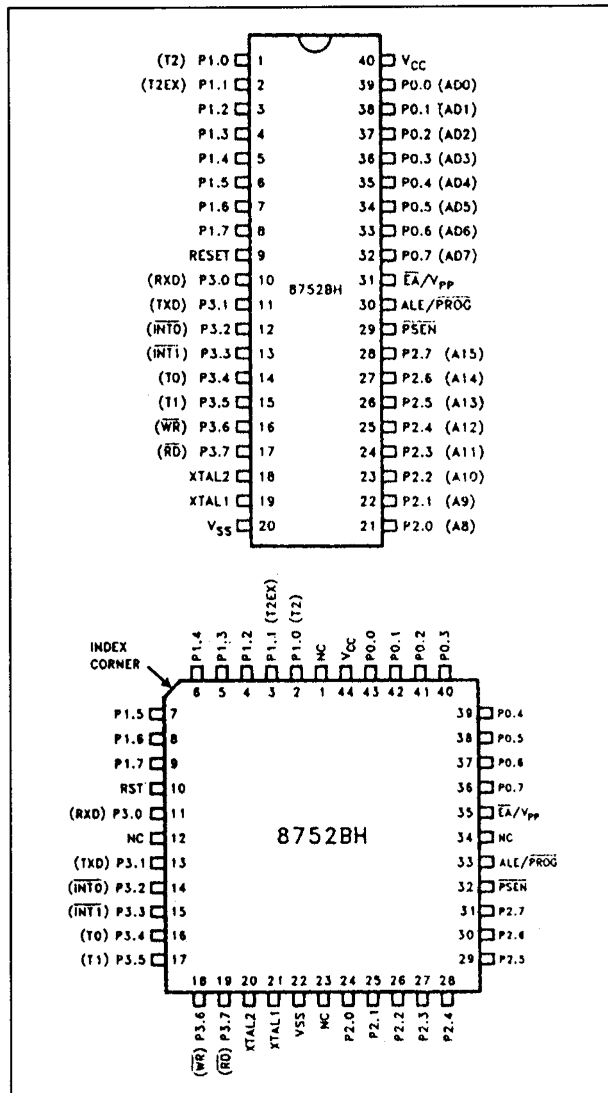
Figuur 7/6.3-21: Opstelling voor het verifiëren van de 8751BH.

De signaturebytes (030H en 031H0) kunnen op dezelfde manier worden uitgelezen als bij een verificatie, maar nu moeten P3.6 en P3.7 LAAG zijn. De waarden zijn:
030H = 89H (fabrikaat: Intel)
031H = 51H (type: 8751BH).

8751BH-beveiliging
De beveiliging (op twee niveaus) bestaat uit twee lockbits en een 32 byte Encryptie Array. Hiermee wordt het programmageheugen beveiligd tegen softwarepiraterij. De EPROM heeft een Encryption Array van 32 bytes, die in het begin niet geprogrammeerd is (allemaal enen). Telkens wanneer een byte bij het verifiëren wordt geadresseerd, worden 5 adreslijnen gebruikt om een byte van het encryptiearray te selecteren. Op deze byte wordt dan de exlusive NOR-functie (XNOR) met de codebyte uitgevoerd, zodat een encrypted verify byte ontstaat. Het wordt aanbevolen dat wanneer het encryptie array wordt gebruikt, tenminste ook één van de lockbits wordt geprogrammeerd.
De functies van de lockbits zijn te zien in tabel 7/6.3-13.

Programmeren van de 8752BH
De 8752BH heeft dezelfde structuur als de 8052, maar dan met 8 kB EPROM aan boord. De 8752BH is verkrijgbaar in drie soorten behuizing: 40-pens plastic of ceramische DIL of 44-pens PLCC.
Alleen de ceramische behuizing is voorzien van een kwartsvenstertje, zodat alleen die versie met UV-licht gewist kan worden (figuur 7/6.3-22).

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



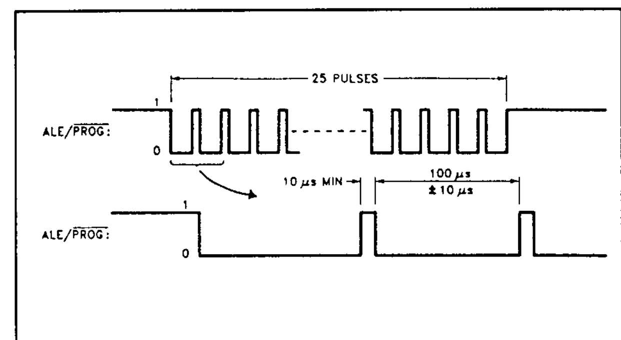
Figuur 7/6.3-22: De 8752BH bevindt zich in een plastic of ceramische DIL-behuizing of in een PLCC. De ceramische DIL heeft een kwarts venster, waardoor de EPROM wisbaar is (de andere zijn OTP: One Time Programmable).

Het programmeren van de 8752BH gaat op dezelfde manier als bij de 8751BH: dezelfde Quick-pulse programmeer-algorithme (figuur 7/6.3-23) en dezelfde programmeerspanning. Alleen is nu een adreslijn meer nodig. De gewenste EPROM-lokatie komt nu op de pennen van poort 1 en P2.0 tot en met P2.4 van poort 2. Voor de programmeermodes wordt verwezen naar tabel 7/6.3-11, terwijl de opstelling in figuur 7/6.3-24 wordt getoond. De elektrische en timing karakteristieken voor het programmeren en verifiëren van de 8752BH zijn identiek aan die in tabel 7/6.3-12 en figuur 7/6.3-20. Ook het verifiëren van de inhoud van de EPROM gebeurt op dezelfde manier als bij de 8751BH (echter met één adreslijn meer: figuur 7/6.3-25). Ook de signaturebytes (030H en 031H0) kunnen net als bij de 8751BH worden uitgelezen. De waarden zijn nu:
030H = 89H (fabrikaat: Intel)
031H = 52H (type: 8752BH).
De beveiliging bestaat (net als bij de 8751BH) uit twee lockbits en een 32 byte encryptie array. De functies van de lockbits zijn ook voor dit type te zien in tabel 7/6.3-13.

Lock Bits		Logic Enabled
LB1	LB2	
U	U	Minimum Program Lock features enabled. (Code Verify will still be encrypted by the Encryption Array)
P	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled
P	P	Same as above, but Verify is also disabled
U	P	Reserved for Future Definition

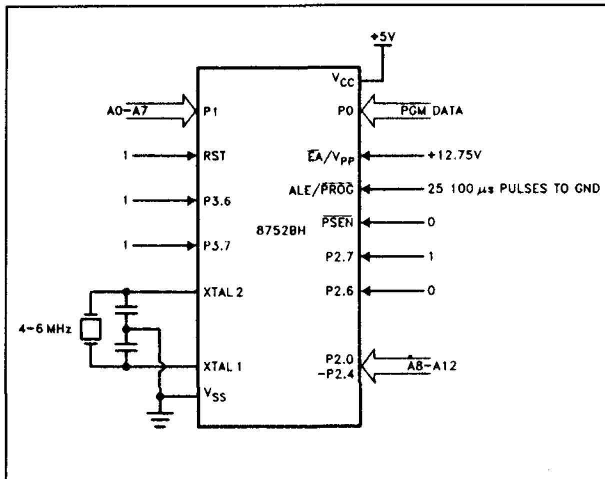
P = Programmed
U = Unprogrammed

Tabel 7/6.3-13: Functies van de lockbits.

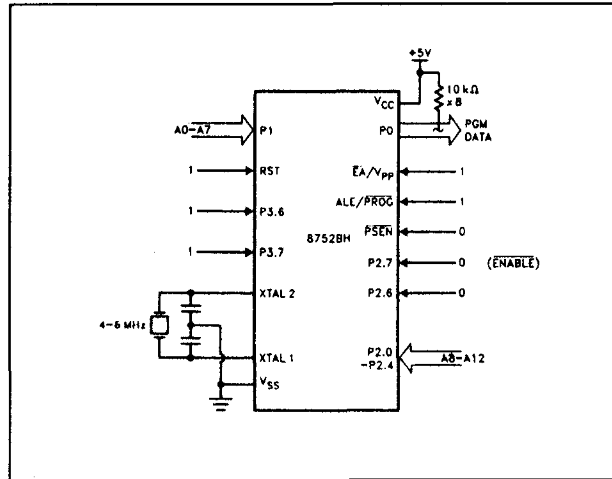


Figuur 7/6.3-23: PROG-golfvormen.

6.3 De basistypen 8031, 8032, 8051, 8052, 8751 en 8752



Figuur 7/6.3-24: Opstelling voor het programmeren van de 8752BH.



Figuur 7/6.3-25: Opstelling voor het verifiëren van de 8752BH.

7/6.4

Overige typen uit de 8051-familie

Algemene gegevens

Inleiding

In dit gedeelte wordt de eerste groep CMOS-typen van de 8051-familie microcontrollers behandeld: de 80C31, 80C51 en 87C51. Alle drie hebben ze dezelfde architectuur als de 8051-familie en zijn daar ook pin-compatibel mee. De belangrijkste kenmerken van deze groep zijn: 128 byte RAM, 2 timer/counters, 32 in-/uitgangspennen en maximaal 4 kB programmageheugen (ROM: 80C51 of EPROM: 87C51 of geen (EP)ROM: 80C31). Bovendien kunnen maximaal 64 kB extern datageheugen worden geadresseerd. Bij alle typen kan gebruik worden gemaakt van de krachtige instructieset van de 8051-familie.

De microcontrollers zijn leverbaar in vier verschillende behuizingen: 40-pens kunststof of ceramische DIL, 44-pens PLCC of 44-pens QFP (zie tabel 7/6.4-1). Het is duidelijk dat alleen de ceramische versie van de 87C51 meerdere keren geprogrammeerd en gewist kan worden.

De 87C51 in de overige behuizingen is slechts éénmaal programmeerbaar (One-Time Programmable: OTP).

Kenmerken

- 8 bit CMOS microcontroller
- 128 byte data-RAM
- Boole'se processor
- 32 in-/uitgangslijnen (vier 8 bit I/O-poorten)
- twee 16 bit timer/counters
- 5 interruptie-bronnen
- 64 kB externe programma geheugenruimte
- 64 kB externe data geheugenruimte
- programmeerbare seriële poort
- TTL- en CMOS-compatibel
- vrijloop en power-down modes
- power-control modes
- snelheden:
  - 80C31BH, 80C51BH, 87C51BH: 3,5 tot 12 MHz
  - 80C31BH-1, 80C51BH-1, 87C51BH-1: 3,5 tot 16 MHz
  - 80C31BH-2, 80C51BH-2, 87C51BH-2: 0,5 tot 12 MHz
  - 87C51-L: 3,5 tot 8 MHz (V_{CC} = 3,3 V +/-0,3 V)
- behuizingen: 40-pens plastic of keramische DIP, 44-pens PLCC of 44-pens QFP (figuren 7/6.4-1, -2 en -3)
- fabrikanten:
  Intel: 80C31BH(-1,-2), 80C51BH(-1,-2), 87C51(-1,-2, -3), 87C51-L, 87C51-20
  Siemens: SAB80C31(-P,-N), SAB80C51(-P,-N)
  Matra-Harris: 80C31(-1), 80C51(-1), 80C31/51-L (2,7 V tot 6 V, 0 tot 6 MHz)
  Philips: SC80C31, SC80C51, SC87C51, P80C51,P80CL31, P80CL51, P85CL000 (Piggy-back versie)
  AMD: 80C31BH, 80C51BH

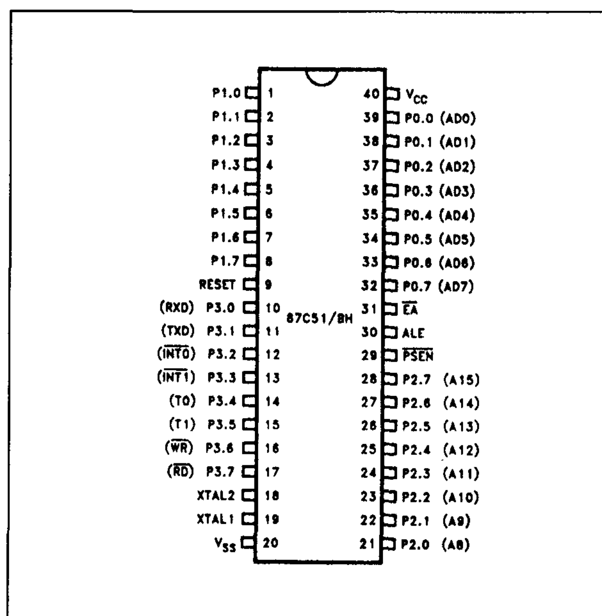
Beschrijving van de aansluitpennen

De nummering van de pennen heeft betrekking op de DIL-uitvoering.

- **V_{CC}, pen 40**
  Voedingsspanning
- **V_{SS}, pen 20**

circuit-aarde

- **Poort 0 (I/O), pennen 32 tot en met 39**
  Poort 0 is een 8 bit open-drain bidirectionele in/uitgangspoort. Elke pen kan bij gebruik als uitgang de stroom voor 8 LSTTL-belastingen "sinken". Poort 0 pennen waarin "enen" zijn geschreven zijn hoog-impedant en kunnen als ingang worden gebruikt. Poort 0 is ook de gemultiplexte lage adres-byte en de databus wanneer toegang wordt verkregen tot extern geheugen. In dat geval worden sterke pull-up's gebruikt om enen te versturen. Bij het programmeren van de EPROM (87C51) ontvangt poort 0 de code-bytes (en brengt die naar buiten tijdens verificatie van het geprogrammeerde).
- **Poort 1 (I/O), pennen 1 tot en met 8**
  Poort 1 is een 8 bit bidirectionele in/uitgangspoort met interne pull-up's. Poort 1 pennen die enen bevatten worden daardoor hoog getrokken en kunnen dan als ingangen worden gebruikt. Poort 1 ontvangt ook de lage adresbytes tijdens het programmeren en verifiëren van de EPROM (87C51).
- **Poort 2 (I/O), pennen 21 tot en met 28**
  Poort 2 is een 8 bit bidirectionele in/uitgangspoort met interne pull-up's. Poort 2 pennen die enen bevatten worden daardoor hoog getrokken en kunnen in die toestand als ingangen worden gebruikt. Poort 2 verstuurt het hoge adres-byte bij de toegang tot extern programmageheugen en bij het adresseren van 16 bit extern datageheugen (MOVX @DPTR). In dat geval worden sterke pull-up's gebruikt bij het verzenden van enen. Worden bij de toegang tot extern datageheugen 8 bit adressen gebruikt (MOVX @Ri), dan verstuurt poort 2 de inhoud van het P2 Special Function Register. Bovendien ontvangt poort 2 de hoge adresbits tijdens het programmeren en verifiëren van de EPROM (8751).
- **Poort 3 (I/O), pennen 10 tot en met 17**
  Poort 3 is een 8 bit bidirectionele in/uitgangspoort met interne pull-up's. Poort 3 pennen waarin enen zijn geschreven, worden daardoor hoog getrokken en kunnen als ingangen worden gebruikt. Poort 3 dient ook voor het uitvoeren van de speciale functies van de 8051-familie, zoals in tabel 7/6.4-2 te zien is. Tevens ontvangt poort 3 enkele besturingssignalen voor het programmeren en verifiëren van de EPROM (87C51).



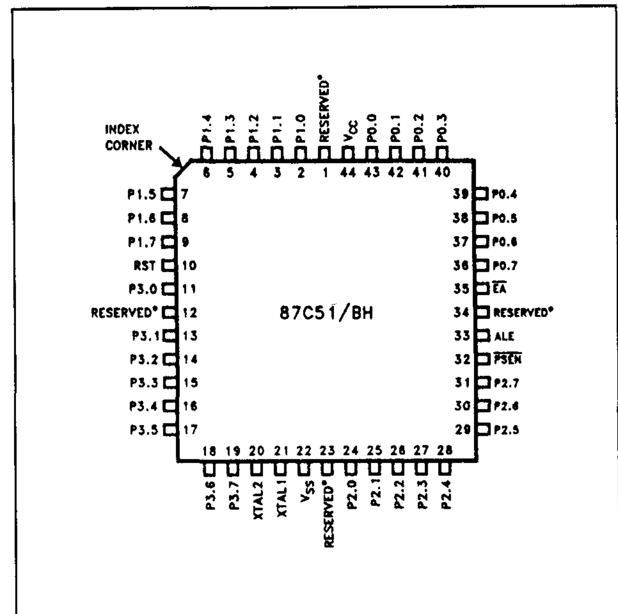
Figuur 7/6.4-1: Aansluitingen van de 40-pens kunststof en ceramische DIL-uitvoering van de 80C31/80C51/87C51. De 87C51 is alleen wis- en herprogrammeerbaar in de keramische DIP-versie.

- **RESET (I), pen 9**
  Wanneer deze pen gedurende twee machine-cycli hoog blijft, terwijl de oscillator draait, wordt de microcontroller gereset. Bij de 87C51 maakt het niet uit of de oscillator al dan niet draait. Door een interne pull-down weerstand kan power-on reset worden bereikt met alleen een externe condensator op pen 9.
- **ALE/$\overline{PROG}$ (O), pen 30**
  Deze uitgang levert het latch enable-signaal om de lage adres-byte in het ex-

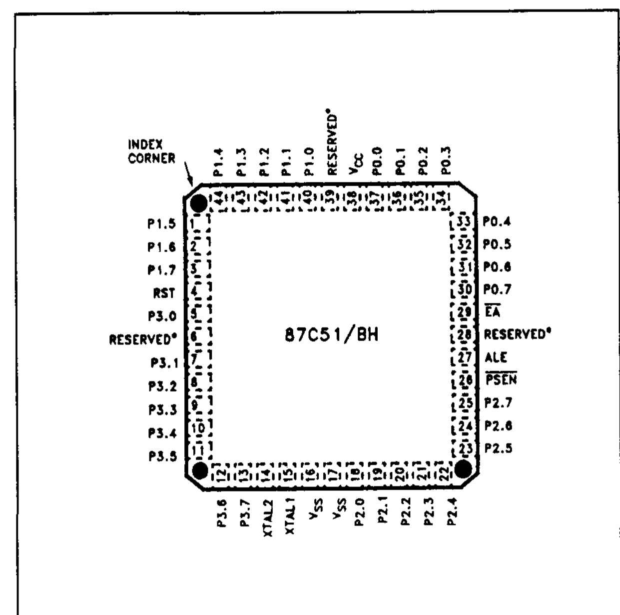
6.4 Overige typen uit de 8051-familie

terne geheugen te latchen. Bij de 87C51 werkt deze pen gebruikt als ingang voor de programmeerpuls voor de EPROM. Indien nodig kan de werking van ALE worden gesperd door bit 0 van de SFR-lokatie 8EH op één te zetten. In dat geval is ALE alleen aktief bij een MOVX of MOVC instructie (anders zwak hoog). Het setten van de ALE disable-bit heeft geen effect als de microcontroller in de externe executiemode werkt. Bij normaal bedrijf wordt ALE met een constante herhalingsfrequentie van 1/6 van de oscillatorfrequentie uitgezonden en kan dan als externe timing worden gebruikt. Let echter op dat bij elke externe datageheugen toegang één ALE-puls wordt overgeslagen.

- **$\overline{\text{PSEN}}$ (O), pen 29**
  Program Store Enable wordt gebruikt om de inhoud van het externe programmageheugen op de bus te zetten. Wanneer de 87C51 met het intern programmageheugen bezig is, is $\overline{\text{PSEN}}$ inactief (hoog). Als de microprocessor bezig is met code uit extern programmageheugen, is $\overline{\text{PSEN}}$ elke tweede machine-cyclus aktief, behalve bij toegang tot extern datageheugen omdat dan twee $\overline{\text{PSEN}}$-aktiveringen worden overgeslagen.
- **$\overline{\text{EA}}$/V_{PP} (I), pen 31**
  Deze external access enable-ingang moet aan V_{SS} worden gelegd om de 87C51 in staat te stellen code op te halen uit de externe programma-adressen tussen 0000H tot en met FFFFH. Voor het uitvoeren van instructies uit het interne geheugen moet $\overline{\text{EA}}$ aan V_{CC} worden gelegd. Deze pen komt bij het programmeren van de EPROM in de 87C51 op 12,75 V te staan.
- **XTAL1, XTAL2, pennen 18 en 19**
  XTAL1 is de ingang van de inverterende oscillator-versterker en XTAL2 de uitgang van de oscillator-versterker. Er kan gebruik gemaakt worden van een kristal (figuur 7/6.4-5) of van een externe clock (figuur 7/6.4-6). In het laatste geval is de aan/uit-verhouding van het signaal onbelangrijk omdat de interne schakeling van een deel-door-twee flip-flop is voorzien.

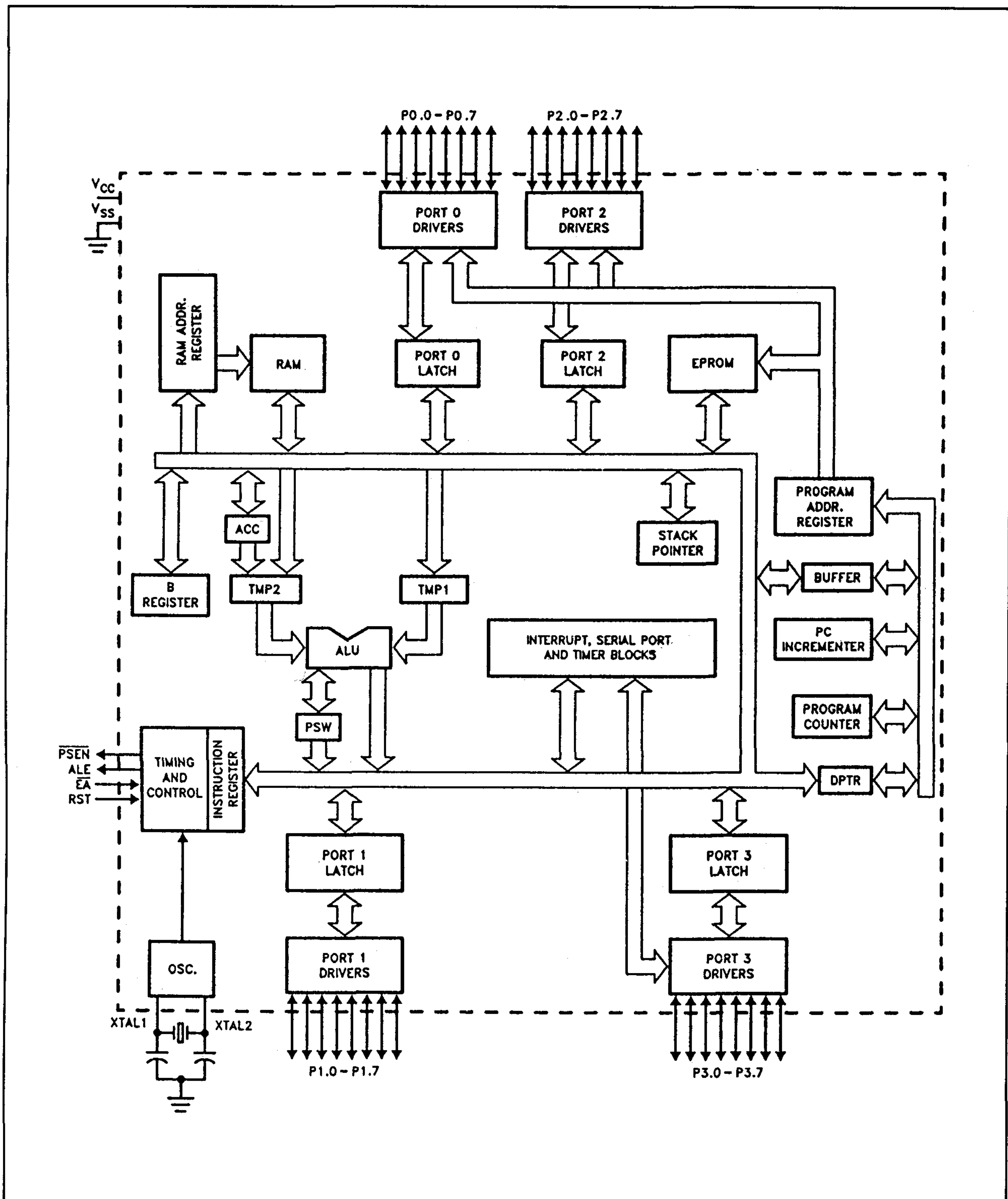


Figuur 7/6.4-2: Aansluitingen van de 80C31/ 80C51/87C51 in de 44-pens kunststof PLCC-uitvoering



Figuur 7/6.4-3: Aansluitingen van de 80C31/ 80C51/87C51 in de 44-pens quad flat pack (QFP) behuizing.

6.4 Overige typen uit de 8051-familie



Figuur 7/6.4-4: Blokschema van de 8031/8051/8751 (bij de 80C51 is de EPROM vervangen door een ROM, terwijl dit gedeelte bij de 80C31 ontbreekt).

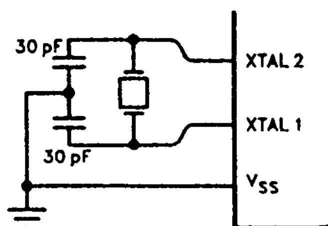
6.4 Overige typen uit de 8051-familie

Part	Prefix	Package Type	θ_{ja}	θ_{jc}
87C51	P	40-Pin Plastic DIP (OTP)	45°C/W	16°C/W
	D	40-Pin CERDIP (EPROM)	45°C/W	15°C/W
	N	44-Pin PLCC (OTP)	46°C/W	16°C/W
	S	44-Pin QFP (OTP)	98°C/W	24°C/W
80C51BH/ 80C31BH	P	40-Pin Plastic DIP	75°C/W	23°C/W
	D	40-Pin CERDIP	36°C/W	13°C/W
	N	44-Pin PLCC	46°C/W	16°C/W
	S	44-Pin QFP	98°C/W	24°C/W

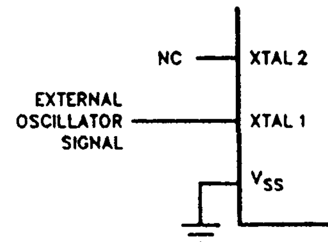
Tabel 7/6.4-1: Overzicht van de beschikbare behuizingen en thermische eigenschappen daarvan.

Pin	Name	Alternate Function
P3.0	RXD	Serial input line
P3.1	TXD	Serial output line
P3.2	$\overline{\text{INT0}}$	External Interrupt 0
P3.3	$\overline{\text{INT1}}$	External Interrupt 1
P3.4	T0	Timer 0 external input
P3.5	T1	Timer 1 external input
P3.6	$\overline{\text{WR}}$	External Data Memory Write strobe
P3.7	$\overline{\text{RD}}$	External Data Memory Read strobe

Tabel 7/6.4-2: Alternatieve functies van de poort 3-pennen.



Figuur 7/6.4-5: Toepassing van de interne oscillator.



Figuur 7/6.4-6: Aansturing met een externe clock.

Vrijloop-mode

In de vrijloop-mode ("idle mode") brengt de microcontroller zichzelf (met software) in slaap, terwijl alle periferie-schakelingen aktief blijven.
De inhouden van de on-chip RAM en de Special Function Registers blijven ongewijzigd.
De vrijloop-mode kan worden beëindigd door elke willekeurige vrijgegeven interrupt of door een hardware reset. Wanneer de vrijloop-mode wordt beëindigd door een hardware reset gaat de microcontroller verder met de uitvoering van het programma vanaf het punt waar hij was gebleven. In dat geval wordt de toegang tot de interne RAM gesperd, maar is wel toegang tot de poortpennen mogelijk.
Om onbedoeld schrijven naar een poort dan te voorkomen mag de instructie die na degene komt die de vrijloop veroorzaakte, niet naar een poort-pen of naar extern geheugen schrijven.

Power down mode

In de power down-mode staat de oscillator stil en is de instructie die hier de oorzaak van was de laatste die werd uitgevoerd. De interne RAM en de Special Function Registers behouden hun waarden. Power down kan alleen worden beëindigd met een hardware reset. Hierdoor worden de SFR's opnieuw gedefinieerd, maar de inhoud van de on-chip RAM verandert niet.

6.4 Overige typen uit de 8051-familie

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

Tabel 7/6.4-3: Status van de externe pennen tijdens vrijloop (idle) en power down

Programmeren van de EPROM

Om de EPROM van de 87C51 te programmeren moet de clockfrequentie 4 tot 6 MHz zijn. Het adres van de te programmeren lokatie wordt op de adreslijnen gezet (poort 1 en een deel van poort 2), terwijl de te programmeren code-byte op de datalijnen (poort 0) wordt gezet. Hierbij moeten de besturings- en programmeersignalen op de niveaus van tabel 7/6.4-9 blijven. Meestal wordt EA/V_{PP} op logisch HOOG gehouden.

Instructieset en overige kenmerken

De instructieset voor deze microcontrollers is reeds opgenomen bij deel 7/6.2. Hieronder volgen de specifieke elektrische en timing-eigenschappen (van de Intel-typen) in de tabellen 7/6.4-4 tot en met -8 en de figuren 7/6.4-7 tot en met -12.

Programmeren van de EPROM

Om de EPROM van de 87C51 te programmeren moet de clockfrequentie 4 tot 6 MHz zijn. Het adres van de te programmeren lokatie wordt op de adreslijnen gezet (poort 1 en een deel van poort 2), terwijl de te programmeren code-byte op de datalijnen (poort 0) wordt gezet. Hierbij moeten de besturings- en programmeersignalen op de niveaus van tabel 7/6.4-9 blijven.
Meestal wordt $\overline{EA}$/V_{PP} op HOOG gehouden tot vlak voordat het programmeren begint.

Ambient Temperature Under Bias . −40°C to +85°C
Storage Temperature .... −65°C to +150°C
Voltage on $\overline{EA}$/V_{PP} Pin to V_{SS} ...... 0V to +13.0V
Voltage on Any Other Pin to V_{SS} .. −0.5V to +6.5V
Maximum I_{OL} per I/O Pin .... 15 mA
Power Dissipation 1.5W
(Based on package heat transfer limitations, not device power consumption).

Tabel 7/6.4-4: De maximaal toegelaten waarden (voor de Intel-typen).

Op dat punt wordt $\overline{EA}$/V_{PP} verhoogd tot V_{PP} (12,75 V) en ALE/$\overline{PROG}$ wordt LAAG gepulst. Daarna wordt $\overline{EA}$/V_{PP} weer HOOG gemaakt (zie ook figuur 7/6.4-13).
Om de 87C51 te programmeren moet de volgende volgorde worden uitgevoerd:
- Zet het juiste adres op de adreslijnen.
- Zet de juiste data op de datalijnen.
- Aktiveer de geschikte combinatie besturingssignalen.
- Verhoog $\overline{EA}$/V_{PP} van V_{CC} tot 12,75 V +/-0,25 V.

6.4 Overige typen uit de 8051-familie

OPERATING CONDITIONS

T_A (under Bias) = 0°C to +70°C; V_{CC} = 5V ±20%; V_{SS} = 0V (87C51-L, V_{CC} = 3.3V ±0.3V)

DC CHARACTERISTICS (Over Operating Conditions)

All parameter values apply to all devices unless otherwise indicated.

Symbol	Parameter	Min	Typ(1)	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5		0.2 V_{CC} − 0.1	V	
V_{IL1}	Input Low Voltage $\overline{EA}$	0		0.2 V_{CC} − 0.3	V	
V_{IH}	Input High Voltage (Except XTAL1, RST)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage (XTAL1, RST)	0.7 V_{CC}		V_{CC} + 0.5	V	
V_{OL}	Output Low Voltage(6) (Ports 1, 2, 3)			0.3	V	I_{OL} = 100 µA(2)
				0.45	V	I_{OL} = 1.6 mA(2)
				1.0	V	I_{OL} = 3.5 mA(2)
V_{OL1}	Output Low Voltage(6) (Port 0, ALE, $\overline{PSEN}$)			0.3	V	I_{OL} = 200 µA(2)
				0.45	V	I_{OL} = 3.2 mA (2)
				1.0	V	I_{OL} = 7.0 mA(2)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$) 87C51	V_{CC} − 0.3			V	I_{OH} = −10 µA(3)
		V_{CC} − 0.7			V	I_{OH} = −30 µA(3)
		V_{CC} − 1.5			V	I_{OH} = −60 µA(3)
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$) 80C51BH/31BH	0.9 V_{CC}			V	I_{OH} = −10 µA(3) V_{CC} = 5V ±10%
		0.75 V_{CC}			V	I_{OH} = −25 µA
		2.4			V	I_{OH} = −60 µA(3)
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode) 87C51	V_{CC} − 0.3			V	I_{OH} = −200 µA(3)
		V_{CC} − 0.7			V	I_{OH} = −3.2 mA(3)
		V_{CC} − 1.5			V	I_{OH} = −7.0 mA(3)
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode) 80C51BH/31BH	0.9 V_{CC}			V	I_{OH} = −80 µA V_{CC} = 5V ±10%
		0.75 V_{CC}			V	I_{OH} = −300 µA
		2.4			V	I_{OH} = −800 µA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3)			−50	µA	V_{IN} = 2V (87C51) V_{IN} = 0.45V
I_{LI}	Input Leakage Current (Port 0)			±10	µA	0.45 < V_{IN} < V_{CC}
I_{TL}	Logical 1-to-0 Transition Current (Ports 1, 2, 3)			−650	µA	V_{IN} = 2V
RRST	RST Pulldown Resistor	50		300	KΩ	
C_{IO}	Pin Capacitance		10		pF	@ 1 MHz, 25°C
I_{CC}	Power Supply Current Active Mode					(Note 5)
	87C51-L at 8 MHz			12	mA	
	All Others at 12 MHz(4)		11.5	20	mA	
	Idle Mode @ 12 MHz(4)		1.7	5	mA	
	Power Down Mode		5	50	µA	

Tabel 7/6.4-5: Gelijkspanningswaarden van de 80C31, 80C51 en 87C51.

6.4 Overige typen uit de 8051-familie

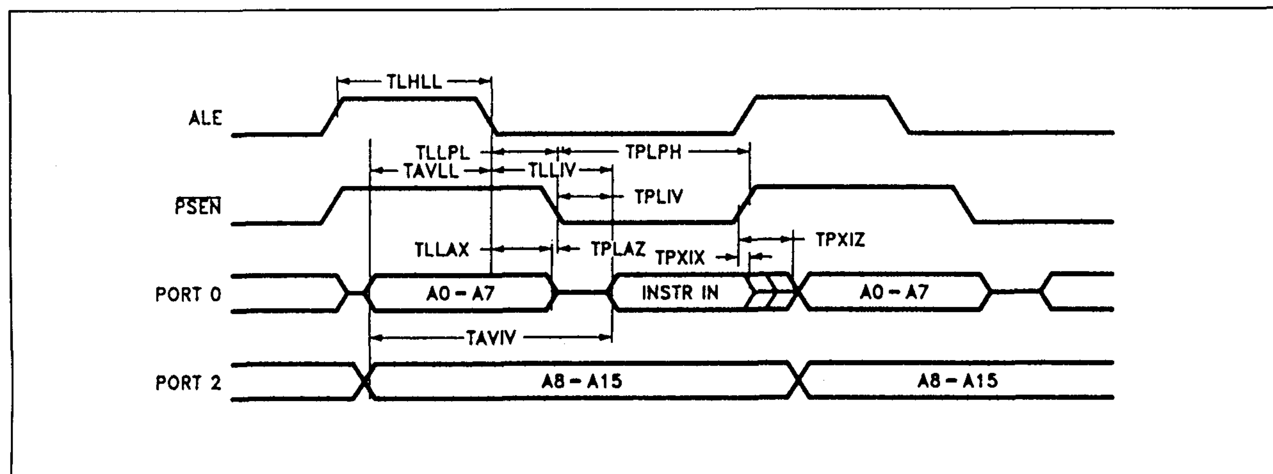
AC CHARACTERISTICS: (Over Operating Conditions; Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF; Load Capacitance for All Other Outputs = 80 pF)

EXTERNAL MEMORY CHARACTERISTICS

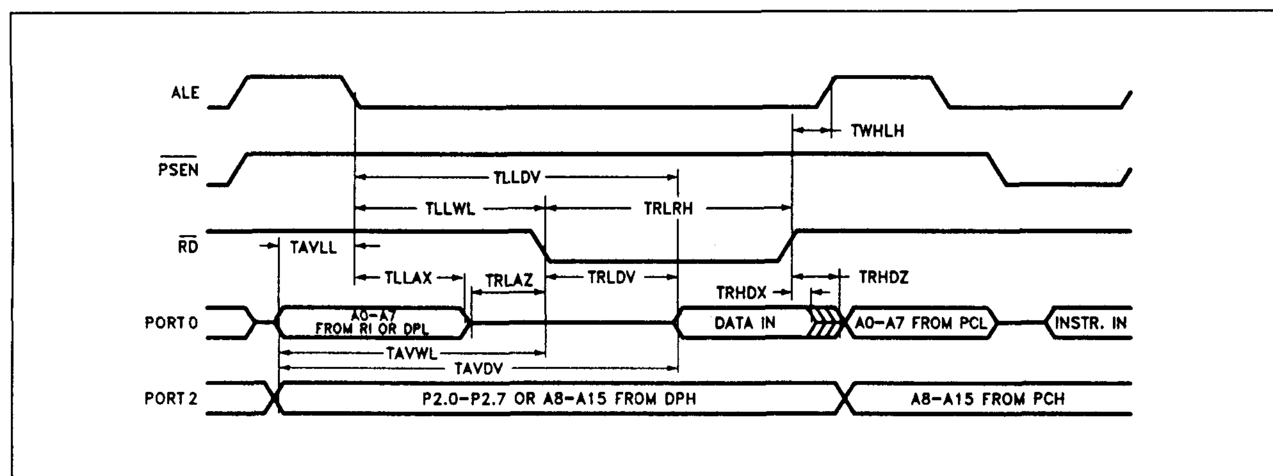
Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency 87C51/BH 87C51-1/BH-1 87C51-2/BH-2			 3.5 3.5 0.5	 12 16 12	 MHz
TLHLL	ALE Pulse Width	127		2TCLCL − 40		ns
TAVLL	Address Valid to ALE Low 87C51 80C51BH/C31BH	 43 28		 TCLCL − 40 TCLCL − 55		 ns ns
TLLAX	Address Hold After ALE Low 87C51 80C51BH/C31BH	 53 48		 TCLCL − 30 TCLCL − 35		 ns ns
TLLIV	ALE Low to Valid Instr In		234		4TCLCL − 100	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low 87C51 80C51BH/C31BH	 53 43		 TCLCL − 30 TCLCL − 40		 ns ns
TPLPH	$\overline{PSEN}$ Pulse Width	205		3TCLCL − 45		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In		145		3TCLCL − 105	ns
TPXIX	Input Instr Hold After $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float After $\overline{PSEN}$		59		TCLCL − 25	ns
TAVIV	Address to Valid Instr In		312		5TCLCL − 105	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL − 100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL − 100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL − 165	ns
TRHDX	Data Hold After $\overline{RD}$	0		0		ns
TRHDZ	Data Float After $\overline{RD}$ 87C51 80C51BH/C31BH		 107 97		 2TCLCL − 60 2TCLCL − 70	 ns ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL − 150	ns
TAVDV	Address to Valid Data In		585		9TCLCL − 165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL − 50	3TCLCL + 50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL − 130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition 87C51 80C51BH/C31BH	 33 23		 TCLCL − 50 TCLCL − 60		 ns ns
TWHQX	Data Hold After $\overline{WR}$	33		TCLCL − 50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	TCLCL − 40	TCLCL + 40	ns

Tabel 7/6.4-6: Timing en schakeltijden.

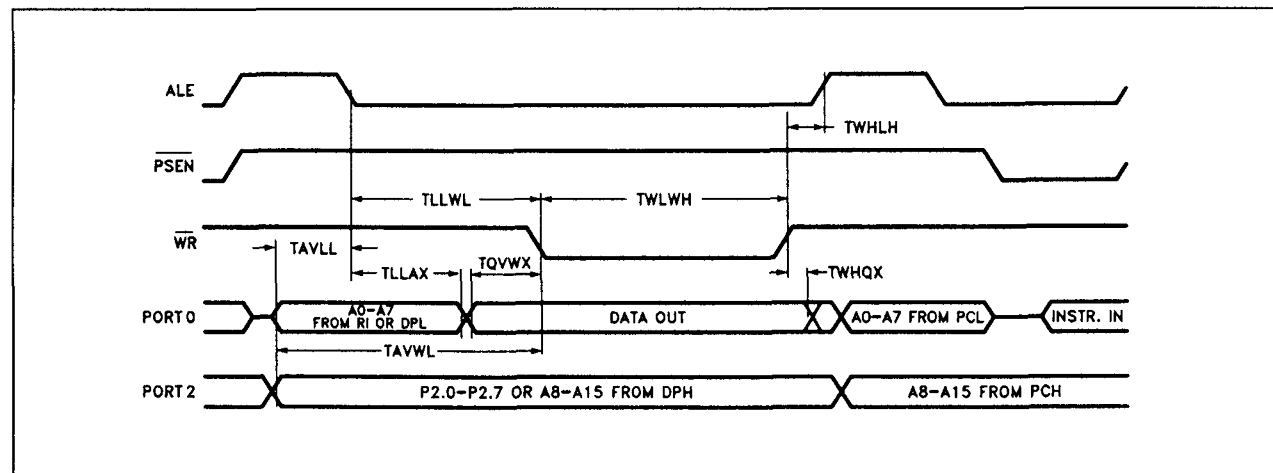
6.4 Overige typen uit de 8051-familie



Figuur 7/6.4-7: Uitlezen van extern programma-geheugen (zie ook tabel 7/6.4-6).



Figuur 7/6.4-8: Uitlezen van extern data-geheugen (zie tabel 7/6.4-6).



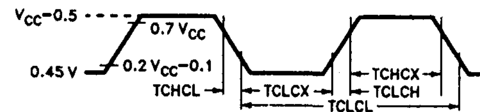
Figuur 7/6.4-9: Schrijven naar extern data-geheugen (zie tabel 7/6.4-6).

6.4 Overige typen uit de 8051-familie

EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency 87C51/BH 87C51-1/BH-1 87C51-2/BH-2	 3.5 3.5 0.5	 12 16 12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

Tabel 7/6.4-7: Timing van de externe clock.

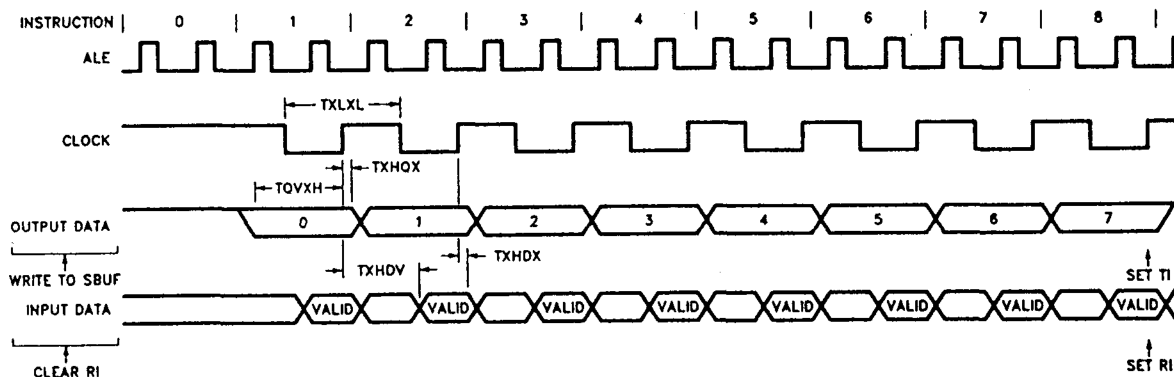


Figuur 7/6.4-10: Golfvormen van het externe clock-signaal (zie tabel 7/6.4-7).

SERIAL PORT TIMING—SHIFT REGISTER MODE

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL − 133		ns
TXHQX	Output Data Hold After Clock Rising Edge	50		2TCLCL − 117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL − 133	ns

Tabel 7/6.4-8: De timing van de seriële poort in de schuifregister-mode.

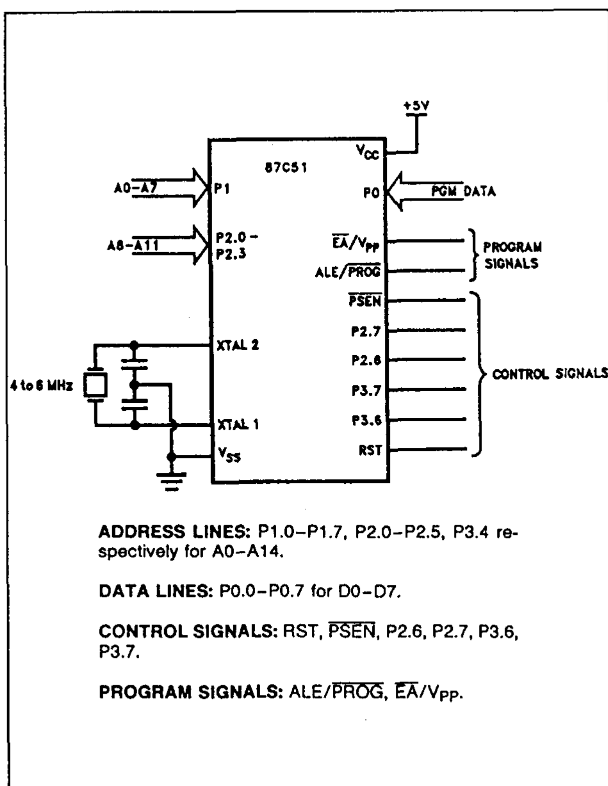


Figuur 7/6.4-11: Golfvormen en schakeltijden bij de schuifregister-mode (zie tabel 7/6.4-8).

6.4 Overige typen uit de 8051-familie

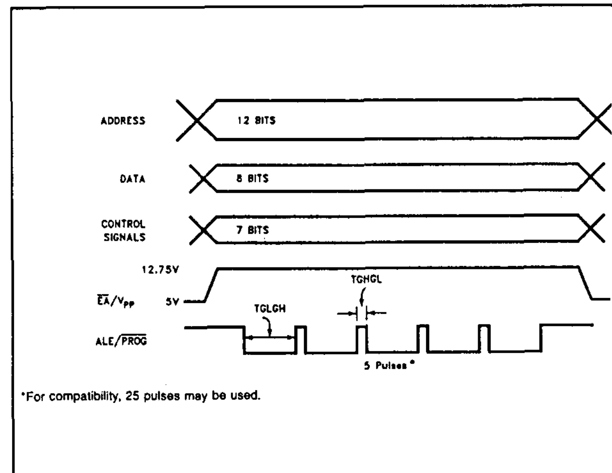
Mode		RST	$\overline{PSEN}$	ALE/ $\overline{PROG}$	$\overline{EA}$/ V_{PP}	P2.6	P2.7	P3.6	P3.7
Program Code Data		H	L	‾|_|‾	12.75V	L	H	H	H
Verify Code Data		H	L	H	H	L	L	H	H
Program Encryption Array Address 0-3F		H	L	‾|_|‾	12.75V	L	H	L	H
Program Lock Bits	Bit 1	H	L	‾|_|‾	12.75V	H	H	H	H
	Bit 2	H	L	‾|_|‾	12.75V	H	H	L	L
	Bit 3	H	L	‾|_|‾	12.75V	H	L	H	L
Read Signature Byte		H	L	H	H	L	L	L	L

Tabel 7/6.4-9: EPROM programmeer-modes (87C51 en 87C52).



Figuur 7/6.4-12: Aansluitingen bij het programmeren van de EPROM.

- Geef 5 pulsen op ALE/$\overline{PROG}$ voor het EPROM-array en 25 pulsen voor de encryptietabel en de lock-bits.
- Herhaal de stappen 1 tot en met 5 bij verschillende adressen en data totdat de gehele array is geprogrammeerd.



Figuur 7/6.4-13: Golfvormen die optreden bij het programmeren.

Tabel 7/6.4-11 bevat de elektrische en timing-karakteristieken voor het programmeren, terwijl de bijbehorende golfvormen in figuur 7/6. te zien zijn.

- Geef 5 pulsen op ALE/$\overline{PROG}$ voor het EPROM-array en 25 pulsen voor de encryptietabel en de lock-bits.
- Herhaal de stappen 1 tot en met 5 bij verschillende adressen en data totdat de gehele array is geprogrammeerd.

Tabel 7/6.4-11 bevat de elektrische en timing-karakteristieken voor het programmeren, terwijl de bijbehorende golfvormen in figuur 7/6. te zien zijn.

Program Lock Bits				Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features enabled. (Code verify will still be encrypted by the encryption array if programmed.)
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, EA is sampled and latched on reset, and further programming of the EPROM is disabled.
3	P	P	U	Same as 2, also verify is disabled.
4	P	P	P	Same as 3, also external execution is disabled.

Tabel 7/6.4-10: Functies van de programmeer-beveiligingsbits.

Programma-verificatie

De EPROM kan worden gecontroleerd na het programmeren van een byte of na een blok van bytes. De beveiligingsbits (Lock bits) kunnen niet direct worden geverifieerd. Deze bits kunnen alleen worden geverifieerd door te bekijken of hun eigenschappen zijn vrijgegeven.

EPROM-beveiliging

De 87C51 heeft een beveiligingssysteem dat, als het geprogrammeerd is, het programma beschermd tegen software-piraterij. De 87C51 heeft een 3 bit “lock”-systeem en een 64 bit encryptie array. Aangezien dit een EPROM-schakeling is, kunnen alle lokaties door de gebruiker worden geprogrammeerd (zie tabel 7/6.4-10).
De encryption array van 64 bytes is in het begin niet geprogrammeerd en bevat allemaal enen. Steeds als een byte bij het verifiëren wordt geadresseerd, worden 6 adreslijnen gebruikt om een byte van het encryptie-array te selecteren. Op dit byte wordt dan een exclusive NOR-functie (XNOR) met het code-byte uitgevoerd, zodat een encrypted verify byte ontstaat.
Wanneer het encryptie-array wordt gebruikt, moet rekening worden gehouden met een belangrijke factor.
Als een code-byte de waarde 0FFH heeft, zal verificatie van dit byte de encryptie byte-waarde opleveren. Als nu een groot deel (>64 bytes) niet geprogrammeerd wordt, zal door een verificatie-routine de inhoud van het encrypte-array zichtbaar worden.
Daarom is het verstandig alle ongebruikte code-bytes te programmeren met een andere waarde dan 0FFH (en niet steeds dezelfde).
Een tweede beveiliging is de “electronic signature” van de 87C51 die onder andere gebruikt wordt voor automatische programmeer-apparatuur (zie tabel 7/6.4-12).

Wissen van de 87C51 (alleen met venster)

De EPROM kan worden gewist door blootstelling aan licht met een golflengte van korter dan 400 nm. Aangezien ook zonlicht en TL-verlichting golflengten in dit gebied hebben, kan blootstelling hieraan (1 week zonlicht of 3 jaar TL) onbedoeld wissen veroorzaken. Het venstertje moet daarom worden bedekt met een niet-doorzichtig stickertje. De aanbevolen wisprocedure is bestraling met UV-licht (253,7 nm) bij een geïntegreerde dosis van minstens 15 $Wsec/cm^2$.

80C32, 80C52 en 87C52

Inleiding

De 80C32, 80C52 en 87C52 vormen de tweede groep CMOS-typen van de 8051-familie. Deze typen hebben 256 bytes RAM en drie 16 bit timer/counters aan boord, terwijl de ROM- en EPROM-versie nu 8 kB programmageheugen bevatten. Verder komen ze overeen met de hiervoor behandelde 80C31/51 en 87C51. Ze hebben dus ook dezelfde architectuur en “pin-out” als de 8051-familie. In dit gedeelte worden daarom

dan ook alleen de afwijkingen ten opzichte van de 80C31, 80C51 en 87C51 behandeld. Om misverstanden te voorkomen (er is immers een teller meer aan boord) worden wel de aansluitingen opgenomen.
De belangrijkste kenmerken van deze groep zijn:
- 256 byte RAM
- 3 timer/counters
- 32 in-/uitgangspennen
- maximaal 8 kB programma-geheugen (ROM: 80C52 of EPROM: 87C52 of geen (EP)ROM: 80C32).

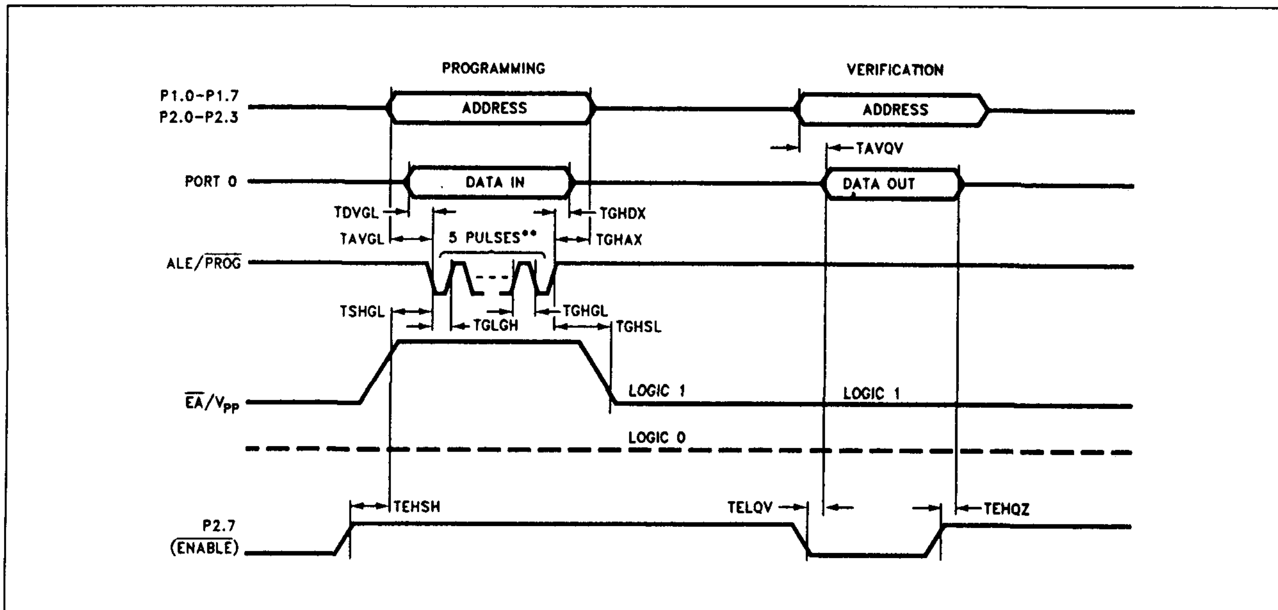
Bovendien kan maximaal 64 kB extern datageheugen worden geadresseerd. Voor alle typen geldt de krachtige instructieset van de 8051-familie. De microcontrollers zijn leverbaar in vier verschillende behuizingen: 40-pens kunststof of ceramische DIL, 44-pens PLCC of 44-pens QFP (zie tabel 7/6.4-13). Het zal duidelijk zijn dat alleen de 87C52 in de ceramische behuizing meerdere keren geprogrammeerd en gewist kan worden. De 87C52 in de andere behuizingen is slechts éénmaal programmeerbaar (One-Time Programmable: OTP).

EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS:
(T_A = 21°C to 27°C, V_{CC} = 5V ±10%, V_{SS} = 0V)

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	12.5	13.0	V
I_{PP}	Programming Supply Current		75	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{PROG}$ Low	48TCLCL		
TGHAX	Address Hold After $\overline{PROG}$	48TCLCL		
TDVGL	Data Setup to $\overline{PROG}$ Low	48TCLCL		
TGHDX	Data Hold After $\overline{PROG}$	48TCLCL		
TEHSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48TCLCL		
TSHGL	V_{PP} Setup to $\overline{PROG}$ Low	10		μs
TGHSL	V_{PP} Hold After $\overline{PROG}$	10		μs
TGLGH	$\overline{PROG}$ Width	90	110	μs
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{ENABLE}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float After $\overline{ENABLE}$	0	48TCLCL	
TGHGL	$\overline{PROG}$ High to $\overline{PROG}$ Low	10		μs

Tabel 7/6.4-11: Voorwaarden voor het programmeren van de EPROM.

6.4 Overige typen uit de 8051-familie



Figuur 7/6.4-14: Golfvormen bij het programmeren en verifiëren van de EPROM in de 87C51.

Location	Contents
	87C51
30H	89H
31H	58H
60H	51H

Tabel 7/6.4-12: Electronische identificatie van de 87C51.

Kenmerken

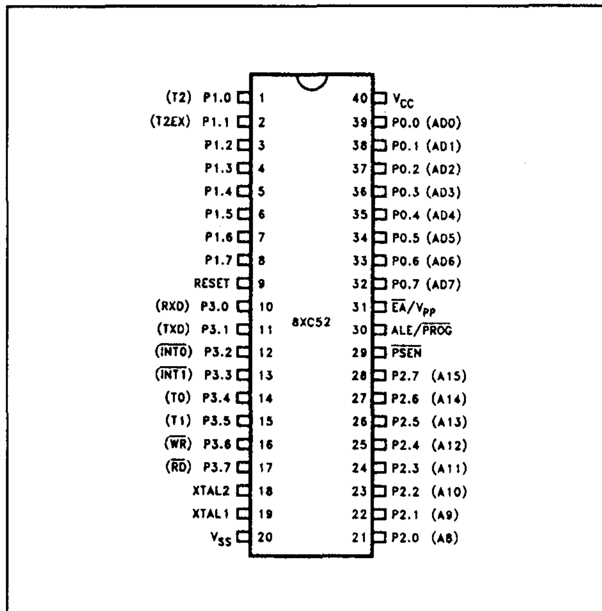
- 8 bit CMOS microcontroller
- 256 byte data-RAM
- 32 kB EPROM/ROM
- Boole'se processor
- 32 in/uitgangslijnen (vier 8 bit I/O-poorten)
- drie 16 bit timer/counters
- 6 interruptie-bronnen
- 64 kB externe programma-geheugenruimte
- 64 kB externe data-geheugenruimte
- programmeerbare seriële poort (met framing error detectie en automatisch adresherkenning)
- TTL- en CMOS-compatibel
- vrijloop en power-down modes
- interrupt-prioriteit op 4 niveaus
- snelheden
  - 80C32, 80C52, 87C52:
    3,5 tot 12 MHz
  - 80C32-1, 80C52-1, 87C52-1:
    3,5 tot 16 MHz
  - 87C52-L:
    3,5 tot 8 MHz (V_{CC} = 3,3 V +/-0,3 V)
- behuizingen: 40-pens plastic of ceramische DIP, 44-pens PLCC of 44-pens QFP (figuren 7/6.4-15, -16 en -17)
- fabrikanten:
  Intel: 80C32(-1,-20), 80C52(-1,-3,-20), 87C52(-1,-3,-20), 87C52-L
  Siemens: SAB80C32(-P,-N), SAB80C52(-P,-N)
  Matra-Harris: 80C32, 80C52
  Philips: P80C32, P80C52, P87C52

Beschrijving van de aansluitpennen

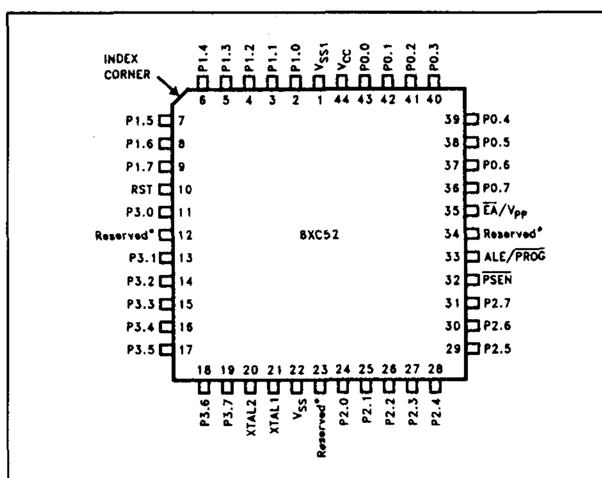
Voor de beschrijving van de aansluitpennen die hier niet worden behandeld wordt verwezen naar de 80C31, 80C51 en 87C51. De

6.4 Overige typen uit de 8051-familie

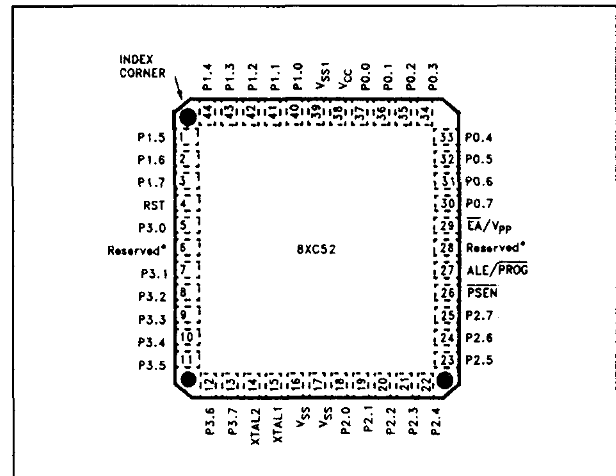
nummering van de pennen heeft betrekking op de DIL-uitvoering.
De werking van de poorten 0, 2 en 3 is gelijk aan die van de 80C31/51 en 87C51 (dus ook tabel 7/6.4-2 gebruiken).



Figuur 7/6.4-15: Aansluitingen van de 40-pens kunststof en keramische DIL-uitvoering van de 80C32/80C52/87C52.



Figuur 7/6.4-16: Aansluitingen van de 80C32/80C52/87C52 in de 44-pens kunststof leadless chip-carrier-uitvoering (PLCC).



Figuur 7/6.4-17: Aansluitingen van de 80C32/80C52/87C52 in de 44-pens quad flat pack (QFP) behuizing.

Part	Prefix	Package Type	θ_{ja}	θ_{jc}
8XC52	P	40-Pin Plastic DIP (OTP)	45°C/W	16°C/W
87C52	D	40-Pin CERDIP (EPROM)	45°C/W	15°C/W
8XC52	N	44-Pin PLCC (OTP)	46°C/W	16°C/W
8XC52	S	44-Pin QFP (OTP)	87°C/W	18°C/W

Tabel 7/6.4-13: Overzicht van de beschikbare behuizingen en thermische eigenschappen daarvan.

- **Poort 1 (I/O), pennen 1 tot en met 8**
  Poort 1 is een 8 bit bidirectionele in-/uitgangspoort met interne pull-up's. Poort 1 ontvangt ook het lage adresbytes tijdens het programmeren en verifiëren van de EPROM (87C52). Daarnaast dienen de pennen 1 en 2 als besturing voor de derde timer/counter T2 (zie tabel 7/6.4-14).

Instructieset en overige kenmerken

De instructieset voor deze microcontrollers is reeds opgenomen bij deel 7/6.2. Het blokschema en de elektrische en timing-eigenschappen van de 80C32, 80C52 en 87C52 komen volledig overeen met die van de 80C31, 80C51 en 87C51.

6.4 Overige typen uit de 8051-familie

Port Pin	Alternate Function
P1.0	T2 (External Count Input to Timer/ Counter 2), Clock-Out
P1.1	T2EX (Timer/Counter 2 Capture/ Reload Trigger and Direction Control)

Tabel 7/6.4-14: Alternatieve functies van P1.0 en P1.1.

Instructieset en overige kenmerken

De instructieset voor deze microcontrollers is reeds opgenomen bij deel 7/6.2. Het blokschema en de elektrische en timing-eigenschappen van de 80C32, 80C52 en 87C52 komen volledig overeen met die van de 80C31, 80C51 en 87C51.

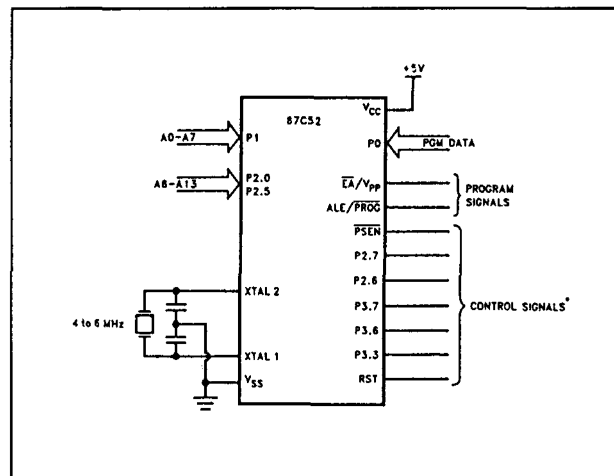
Programmeren van de EPROM

Om de EPROM van de 87C52 te programmeren moet de clockfrequentie 4 tot 6 MHz zijn. Het adres van de te programmeren lokatie wordt op de adreslijnen gezet (poort 1 en een deel van poort 2), terwijl het te programmeren code-byte op de datalijnen (poort 0) wordt gezet, zie figuur 7/6.4-18. Hierbij moeten de besturings- en programmeersignalen op de niveaus van tabel 7/6.4-9 blijven. Meestal wordt $\overline{EA}/V_{PP}$ HOOG gehouden tot vlak voordat het programmeren begint. Dan wordt $\overline{EA}/V_{PP}$ verhoogd tot V_{PP} (12,75 V) en ALE/$\overline{PROG}$ wordt LAAG gepulst. Daarna wordt $\overline{EA}/V_{PP}$ weer HOOG gemaakt (zie ook de figuren 7/6.4-19 en -20). Hierbij geldt hetzelfde programmeer-algorithme als bij de 87C51 (zie daar).

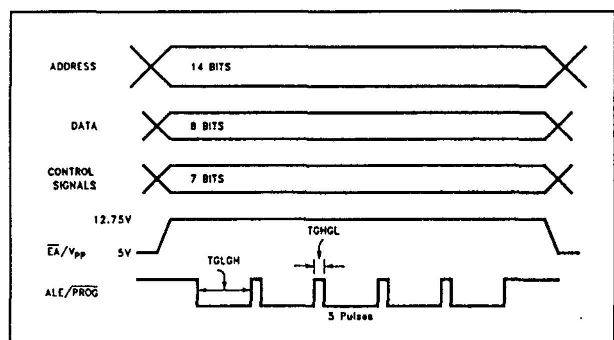
80C54 en 87C54

Inleiding

De derde uitbreiding op de CMOS-versies van de 8051-familie wordt gevormd door de 80C54 en 87C54. Deze typen zijn weer een maat groter en hebben behalve 256 byte RAM en drie 16 bit timer/counters 16 kB programma-geheugen aan boord, in de vorm van ROM of EPROM. Een ROM-loze versie hiervan is er niet. Verder komen ze overeen met de hiervoor behandelde 80C52 en 87C52.



Figuur 7/6.4-18: Aansluiting van de 87C52 voor het programmeren.



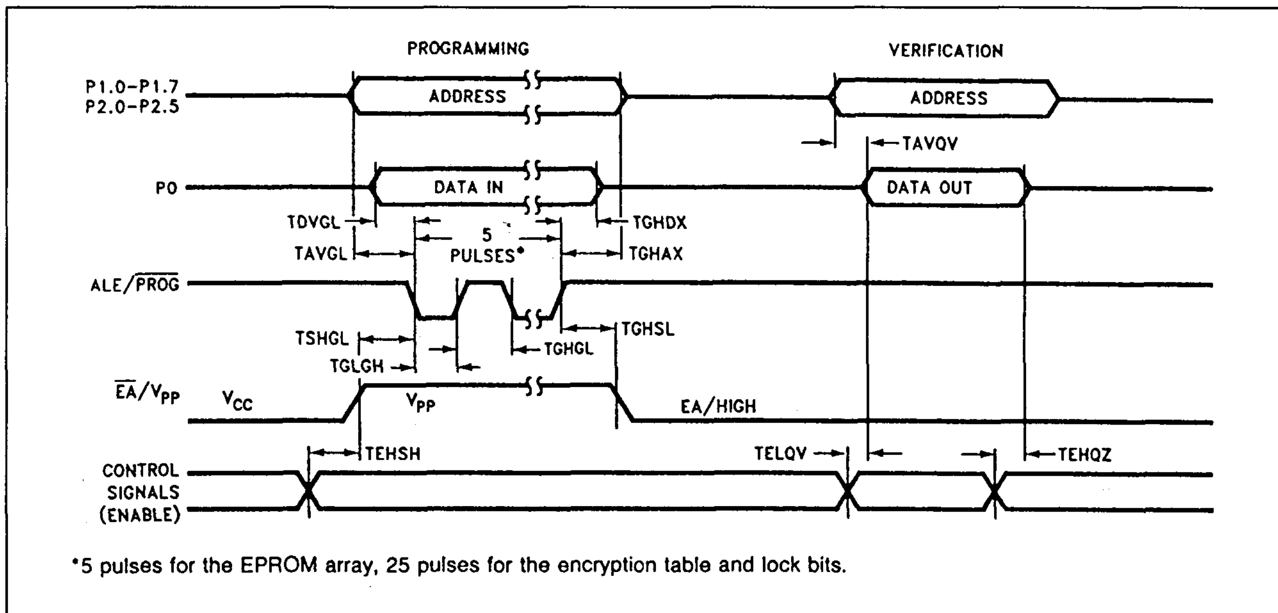
Figuur 7/6.4-19: Golfvormen die optreden bij het programmeren.

In dit gedeelte worden alleen de afwijkingen ten opzichte van de 80C51 en 87C51 behandeld. Wel zijn weer de aansluitingen opgenomen om misverstanden te voorkomen.
De belangrijkste kenmerken van deze groep zijn:

- 256 byte RAM
- 3 timer/counters
- 32 in/uitgangspennen
- 16 kB programmageheugen (80C54: ROM, 87C54: EPROM).

Bovendien kan maximaal 64 kB extern datageheugen worden geadresseerd.

6.4 Overige typen uit de 8051-familie



Figuur 7/6.4-20: Golfvormen bij het programmeren en verifiëren van de EPROM in de 87C52.

Ook nu geldt weer de krachtige instructieset van de 8051-familie. De microcontrollers zijn leverbaar in vier verschillende behuizingen: 40-pens kunststof of keramische DIL, 44-pens PLCC of 44-pens QFP (zie tabel 7/6.4-16). Het zal duidelijk zijn dat alleen de 87C54 in de ceramische behuizing herprogrammeerbaar is. De 87C54 in de andere behuizingen is slechts éénmaal programmeerbaar (One-Time Programmable: OTP).

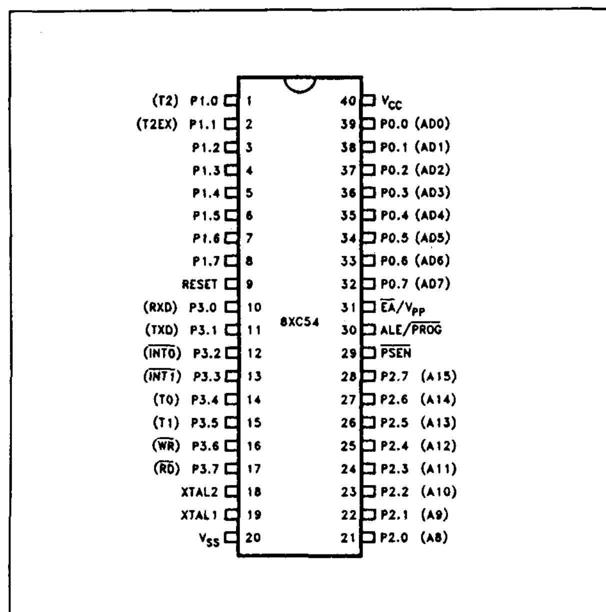
Location	Content	
	87C52	80C52
30H	89H	89H
31H	58H	58H/53H
60H	52H	52H/12H

Tabel 7/6.4-15: Electronische identificatie van de 80C52 en 87C52.

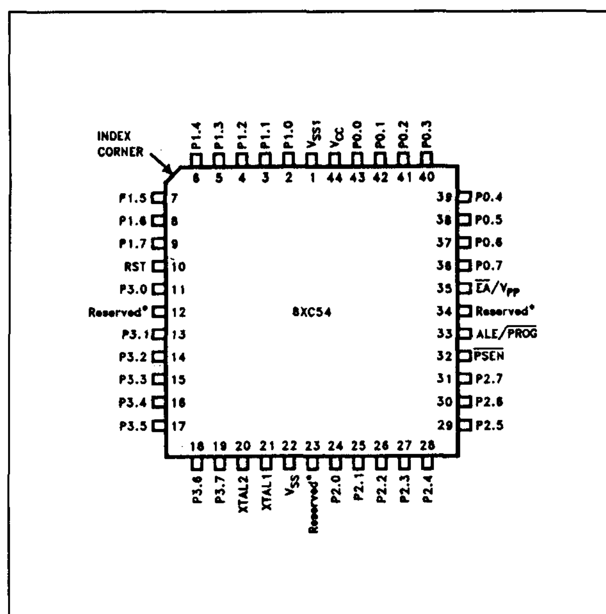
Kenmerken

- 8 bit CMOS microcontroller
- 256 byte data-RAM
- 16 kB EPROM/ROM
- Boole'se processor
- 32 in/uitgangslijnen (vier 8 bit I/O-poorten)
- drie 16 bit timer/counters
- 6 interruptie-bronnen
- 64 kB externe programma-geheugenruimte
- 64 kB externe data-geheugenruimte
- programmeerbare seriële poort (met framing error detectie en automatisch adresherkenning)
- TTL- en CMOS-compatibel
- vrijloop en power-down modes
- interrupt-prioriteit op 4 niveaus
- snelheden
  - 80C54, 87C54: 3,5 tot 12 MHz
  - 80C54-1, 87C54-1: 3,5 tot 16 MHz
  - 80C54-L, 87C54-L: 3,5 tot 8 MHz (V_{CC} = 3,3 V +/-0,3 V)
- behuizingen: 40-pens plastic of ceramische DIP, 44-pens PLCC of 44-pens QFP (figuren 7/6.4-21, -22 en -23)
- fabrikanten:
  Intel: 80C54(-1,-3,-L,-20), 87C54(-1,-3,-L,-20)
  Philips: P80C54, P87C54

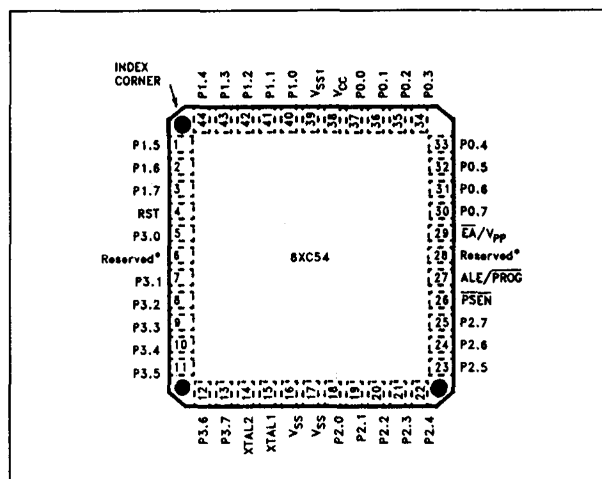
6.4 Overige typen uit de 8051-familie



Figuur 7/6.4-21: Aansluitingen van de 40-pens kunststof en ceramische DIL-uitvoering van de 80C54/87C54. Alleen de 87C54 in de ceramische versie kan worden gewist en geprogrammeerd.



Figuur 7/6.4-22: Aansluitingen van de 80C54/87C54 in de 44-pens kunststof leadless chip-carrier-uitvoering (PLCC).



Figuur 7/6.4-23: Aansluitingen van de 80C54/87C54 in de 44-pens quad flat pack (QFP) behuizing.

Part	Prefix	Package Type	θ_{ja}	θ_{jc}
8XC54	P	40-Pin Plastic DIP (OTP)	45°C/W	16°C/W
87C54	D	40-Pin CERDIP (EPROM)	45°C/W	15°C/W
8XC54	N	44-Pin PLCC (OTP)	46°C/W	16°C/W
8XC54	S	44-Pin QFP (OTP)	96°C/W	24°C/W

Tabel 7/6.4-16: Overzicht van de beschikbare behuizingen en thermische eigenschappen daarvan.

Beschrijving van de aansluitpennen

Voor de beschrijving van de aansluitpennen die hier niet zijn behandeld, wordt verwezen naar de 80C51 en 87C51. De nummering van de pennen heeft betrekking op de DIL-uitvoering. De werking van de poorten 0, 2 en 3 is gelijk aan die van de 80C51 en 87C51 (dus ook tabel 7/6.4-2 gebruiken).

- **Poort 1 (I/O), pennen 1 tot en met 8**
  Poort 1 is een 8 bit bidirectionele in/uitgangspoort met interne pull-up's. Poort 1 ontvangt ook de lage adresbytes tijdens het programmeren en verifiëren van de EPROM (87C54). Tevens dienen de pennen 1 en 2 als besturing voor de derde timer/counter T2 (zie tabel 7/6.4-17).

6.4 Overige typen uit de 8051-familie

Port Pin	Alternate Function
P1.0	T2 (External Count Input to Timer/ Counter 2), Clock-Out
P1.1	T2EX (Timer/Counter 2 Capture/ Reload Trigger and Direction Control)

Tabel 7/6.4-17: Alternatieve functies van P1.0 en P1.1.

Instructieset en overige kenmerken

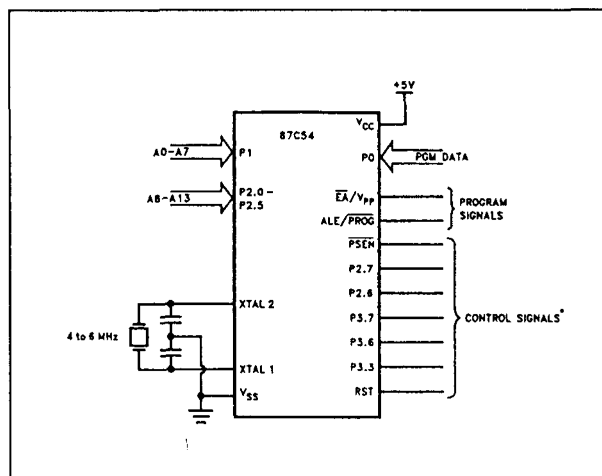
De instructieset voor deze microcontrollers is reeds opgenomen bij deel 7/6.2. Het blokschema en de elektrische en timing-eigenschappen van de 80C54 en 87C54 komen volledig overeen met die van de 80C51 en 87C51.

Programmeren van de EPROM

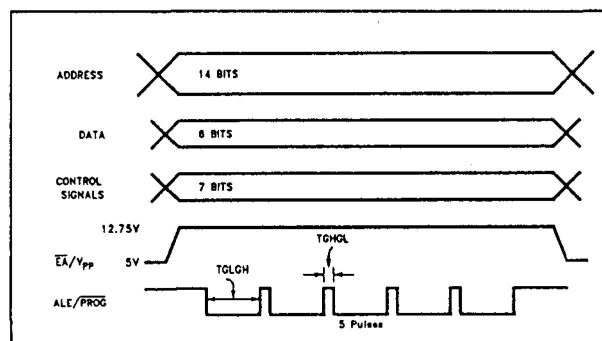
Om de EPROM van de 87C54 te programmeren moet de clockfrequentie 4 tot 6 MHz zijn. Het adres van de te programmeren lokatie wordt op de adreslijnen gezet (poort 1 en een deel van poort 2), terwijl het te programmeren code-byte op de datalijnen (poort 0) wordt gezet, zie figuur 7/6.4-24. Hierbij moeten de besturings- en programmeersignalen op de niveaus van tabel 7/6.4-9 blijven. Meestal wordt $\overline{EA}/V_{pp}$ HOOG gehouden tot vlak voordat het programmeren begint. Dan wordt $\overline{EA}/V_{pp}$ op V_{pp} (12,75 V) gebracht en wordt ALE/$\overline{PROG}$ LAAG gepulst. Daarna wordt $\overline{EA}/V_{pp}$ weer HOOG gemaakt (zie ook de figuren 7/6.4-25 en -26). Hierbij geldt hetzelfde programmeeralgorithme als bij de 87C51 (zie aldaar).

Location	Content	
	87C54	80C54
30H	89H	89H
31H	58H	58H
60H	54H	54H/14H

Tabel 7/6.4-18: Electronische identificatie van de 80C54 en 87C54.



Figuur 7/6.4-24: Aansluiting van de 87C54 voor het programmeren.



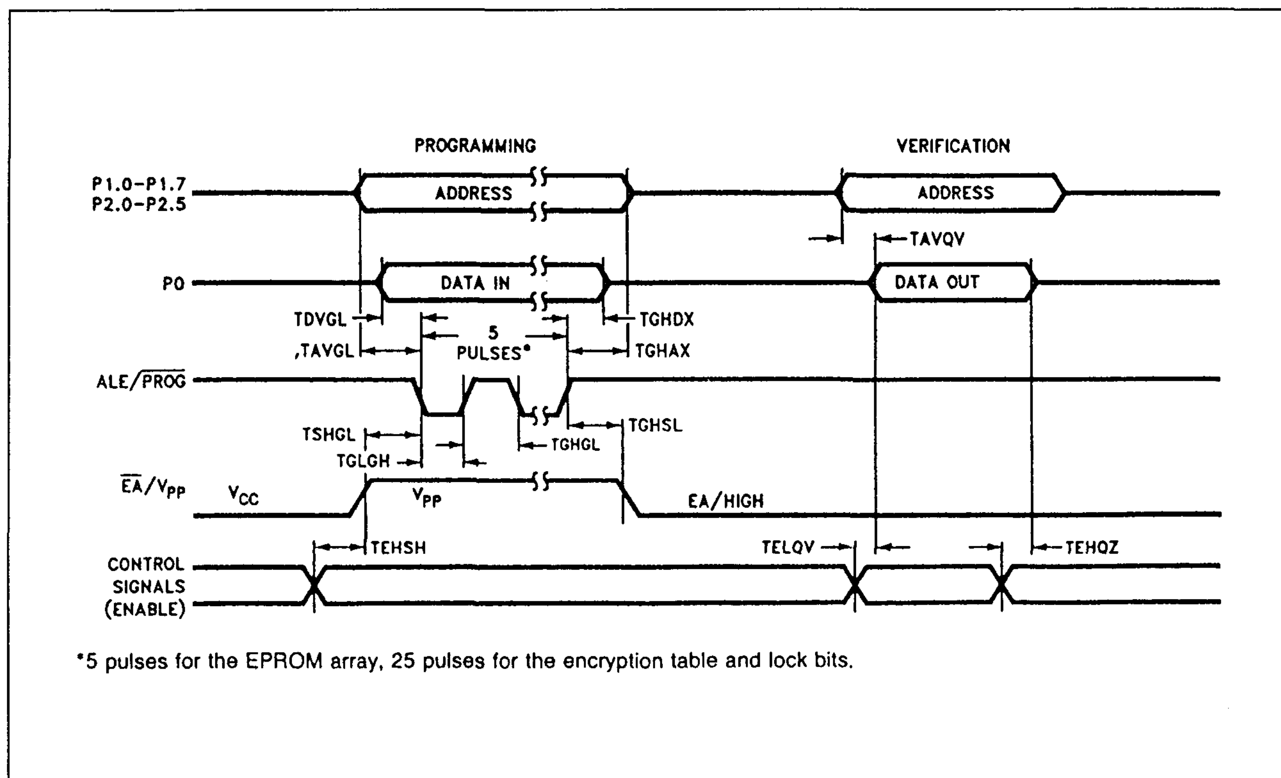
Figuur 7/6.4-25: Golfvormen die optreden bij het programmeren.

80C58 en 87C58

Inleiding

De vierde CMOS-uitbreiding op de 8051-familie bestaat uit de 80C58 en 87C58. Deze typen hebben behalve 256 byte RAM en drie 16 bit timer/counters 32 kB programmageheugen aan boord (ROM of EPROM). Ook hiervan bestaat geen ROM-loze versie. Verder komen ze overeen met de hiervoor behandelde 80C54 en 87C54. Hier worden dus alleen de afwijkingen ten opzichte van de 80C51 en 87C51 behandeld, maar zijn wel weer de aansluitingen opgenomen om misverstanden te voorkomen.

6.4 Overige typen uit de 8051-familie



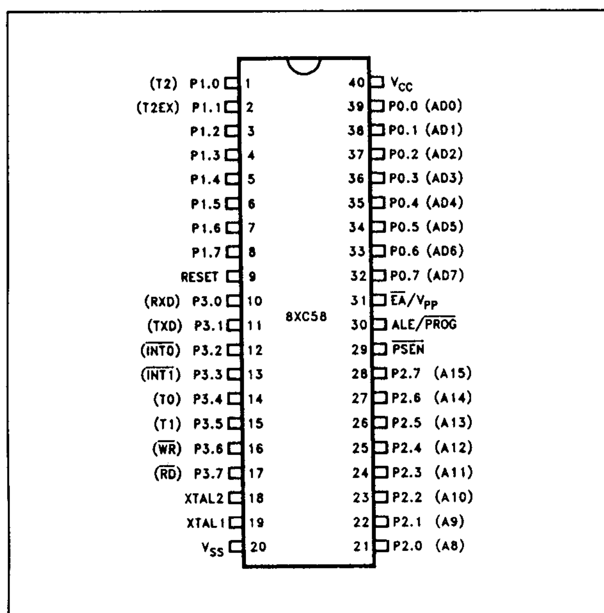
Figuur 7/6.4-26: Golfvormen bij het programmeren en verifiëren van de EPROM in de 87C54.

De belangrijkste kenmerken van deze groep zijn:
- 256 byte RAM
- 3 timer/counters
- 32 in/uitgangspennen
- 32 kB programmageheugen (80C58: ROM, 87C58: EPROM).

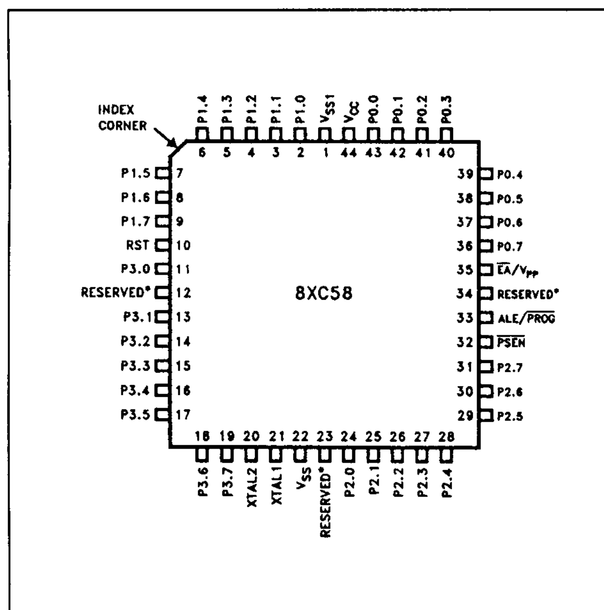
Bovendien kunnen maximaal 64 kB extern datageheugen worden geadresseerd. Ook voor deze typen geldt de krachtige instructieset van de 8051-familie, terwijl ze leverbaar zijn in vier verschillende behuizingen: 40-pens kunststof of ceramische DIL, 44-pens PLCC of 44-pens QFP (zie tabel 7/6.4-19). Alleen de 87C58 in de ceramische behuizing is wis- en herprogrammeerbaar. De 87C58 in een andere behuizing is slechts éénmaal programmeerbaar (One-Time Programmable: OTP).

Kenmerken
- 8 bit CMOS microcontroller
- 256 byte data-RAM
- 32 kB EPROM/ROM
- Boole'se processor
- 32 in/uitgangslijnen (vier 8-bit I/O-poorten)
- drie 16 bit timer/counters
- 6 interruptie-bronnen
- 64 kB externe programma-geheugenruimte
- 64 kB externe data-geheugenruimte
- programmeerbare seriële poort (met framing error detectie en automatisch adresherkenning)
- TTL- en CMOS-compatibel
- vrijloop en power-down modes
- interrupt-prioriteit op 4 niveaus
- snelheden:
  80C58, 87C58: 3,5 tot 12 MHz
  80C58-1, 87C58-1: 3,5 tot 16 MHz
  80C58-L, 87C58-L: 3,5 tot 8 MHz (V_{CC} = 3,3 V +/-0,3 V)
- behuizingen: 40-pens plastic of ceramische DIP, 44-pens PLCC of 44-pens QFP (figuren 7/6.4-27, -28 en -29)
- fabrikanten:

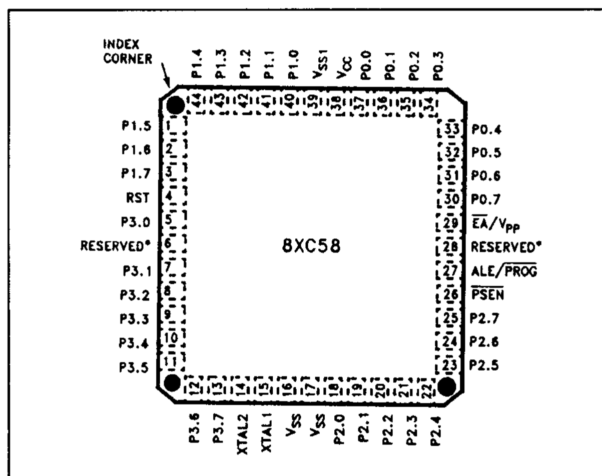
6.4 Overige typen uit de 8051-familie



Figuur 7/6.4-27: Aansluitingen van de 40-pens kunststof en ceramische DIL-uitvoering van de 80C58/87C58. Alleen de 87C58 in de ceramische versie kan worden gewist en geprogrammeerd.



Figuur 7/6.4-28: Aansluitingen van de 80C58/87C58 in de 44-pens kunststof leadless chip-carrier-uitvoering (PLCC).



Figuur 7/6.4-29: Aansluitingen van de 80C58/87C58 in de 44-pens quad flat pack (QFP) behuizing.

Part	Prefix	Package Type	θ_{ja}	θ_{jc}
8XC58	P	40-Pin Plastic DIP (OTP)	45°C/W	16°C/W
87C58	D	40-Pin CERDIP (EPROM)	45°C/W	15°C/W
8XC58	N	44-Pin PLCC (OTP)	46°C/W	16°C/W
8XC58	S	44-Pin QFP (OTP)	90°C/W	22°C/W

Tabel 7/6.4-19: Overzicht van de beschikbare behuizingen en thermische eigenschappen daarvan.

- Intel: 80C58 (-1, -3, -L, -20), 87C58 (-1,-3, -L, -20)
- Philips: P80C58, P87C58

Beschrijving van de aansluitpennen

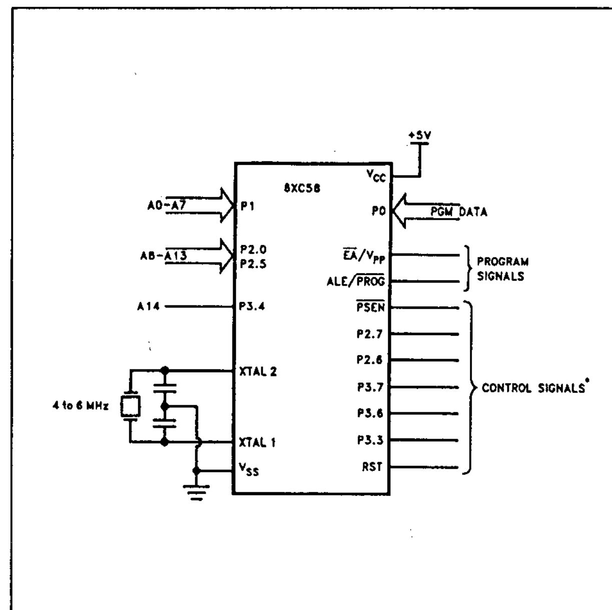
Voor de beschrijving van de aansluitpennen wordt verwezen naar de 80C51 en 87C51. De nummering van de pennen heeft betrekking op de DIL-uitvoering. De werking van de poorten 0, 2 en 3 is gelijk aan die van de 80C51 en 87C51 (dus ook tabel 7/6.4-2 gebruiken).

- **Poort 1 (I/O), pennen 1 tot en met 8**
  Poort 1 is een 8 bit bidirectionele in-/uitgangspoort met interne pull-up's. Poort 1

ontvangt ook de lage adresbytes tijdens het programmeren en verifiëren van de EPROM (87C58). Tevens dienen de pennen 1 en 2 als besturing voor de derde timer/counter T2 (zie tabel 7/6.4-20).

Port Pin	Alternate Function
P1.0	T2 (External Count Input to Timer/Counter 2), Clock-Out
P1.1	T2EX (Timer/Counter 2 Capture/Reload Trigger and Direction Control)

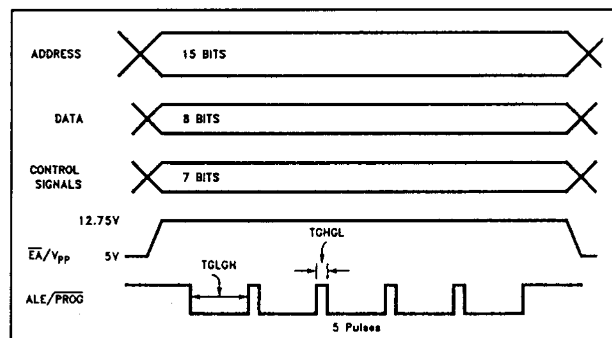
Tabel 7/6.4-20: Alternatieve functies van P1.0 en P1.1.



Figuur 7/6.4-30: Aansluiting van de 87C58 om te kunnen programmeren.

Instructieset en overige kenmerken

De instructieset voor deze microcontrollers is reeds opgenomen bij deel 7/6.2. Het blokschema en de elektrische en timing-eigenschappen van de 80C58 en 87C58 komen volledig overeen met die van de 80C51 en 87C51.



Figuur 7/6.4-31: Golfvormen die optreden bij het programmeren.

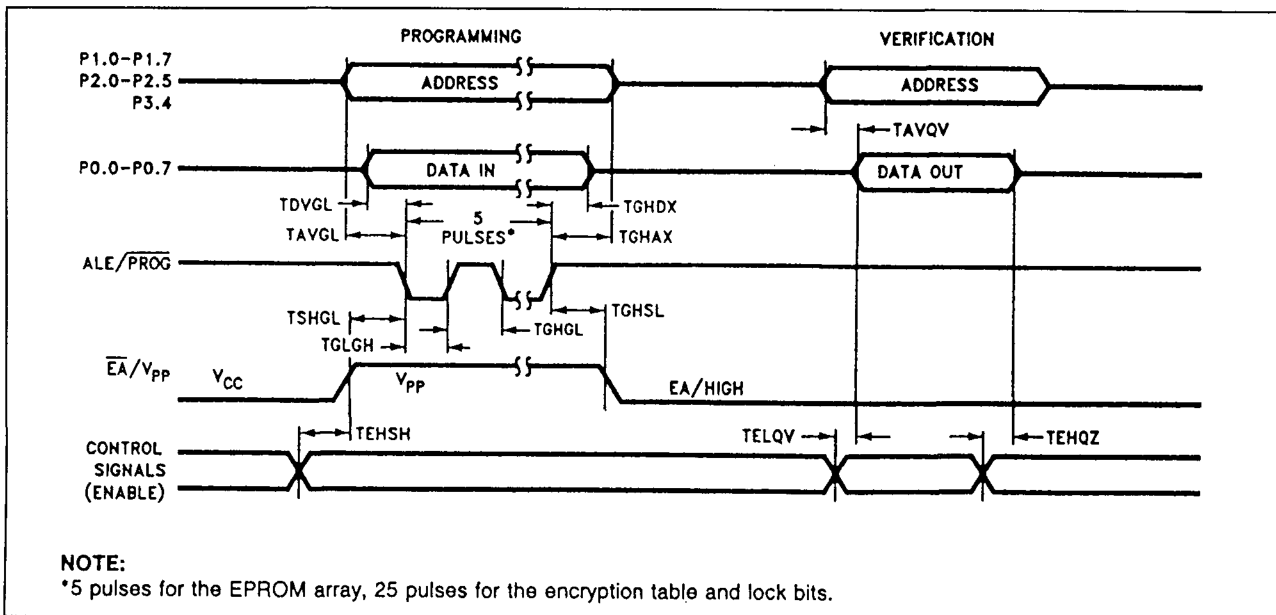
Programmeren van de EPROM

Om de EPROM in de 87C58 te programmeren moet de clockfrequentie 4 tot 6 MHz zijn. Het adres van de te programmeren lokatie wordt op de adreslijnen gezet (poort 1 en een deel van poort 2), terwijl het te programmeren code-byte op de datalijnen (poort 0) wordt gezet, zie figuur 7/6.4-30. Hierbij moeten de besturings- en programmeersignalen op de niveaus van tabel 7/6.4-9 blijven. Meestal wordt $\overline{EA}/V_{PP}$ HOOG gehouden tot het programmeren begint. Dan wordt $\overline{EA}/V_{PP}$ naar V_{PP} (12,75V) verhoogd en wordt ALE/$\overline{PROG}$ LAAG gepulst. Daarna wordt $\overline{EA}$/Vpp weer HOOG gemaakt (zie ook de figuren 7/6.4-31 en -32). Hierbij geldt hetzelfde programmeer-algorithme als bij de 87C51 (zie aldaar).

Location	Content	
	87C58	80C58
30H	89H	89H
31H	58H	58H
60H	58H	58H/18H

Tabel 7/6.4-21: Electronische identificatie van de 80C58 en 87C58.

6.4 Overige typen uit de 8051-familie



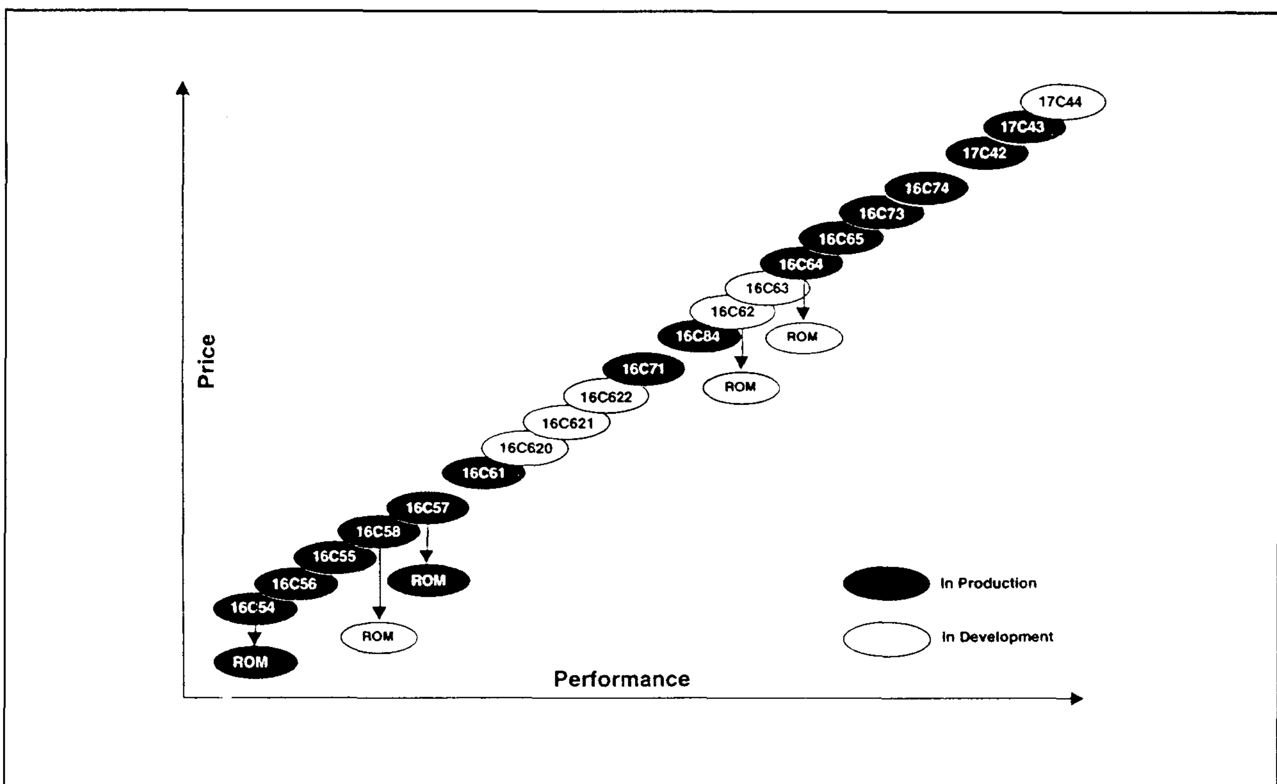
Figuur 7/6.4-32: Golfvormen bij het programmeren en verifiëren van de EPROM in de 87C58.

6.4 Overige typen uit de 8051-familie

7/6.5.1

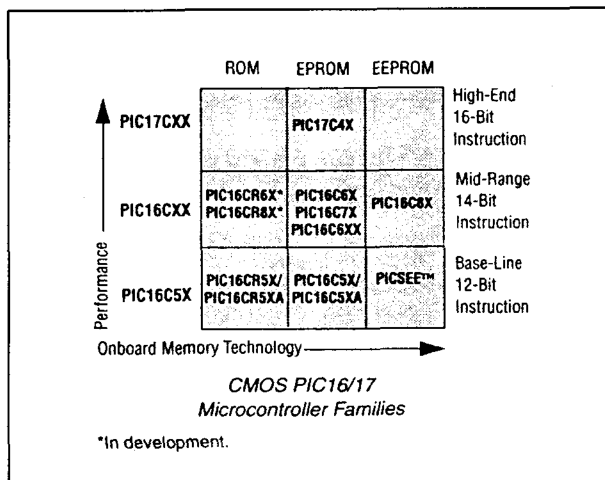
Achtergrond-informatie van de PIC16/17-familie

Inleiding
De PIC16/17 microcontrollers van de firma Microchip zijn door hun hoge prestaties, lage prijs en kleine behuizing een goede keus voor toepassingen in allerlei consumentenprodukten, computer-randapparatuur, kantoormachines, beveiligings-installaties en telecommunicatie systemen.
Deze microcontrollers worden met allerlei configuraties gemaakt en vormen zo een hechte familie van elkaar logisch aanvullende typen (zie figuur 7/6.5.1-1). Bovendien zijn dit de enige 8 bit microcontrollers die gebruik maken van een snelle RISC-architectuur (Reduced Instruction Set) om de snelheid en de efficiency te verbeteren.
Op dit moment zijn drie families 8 bit microcontrollers leverbaar:
- PIC16C5x: de basisklasse
- PIC16Cxx: de middenklasse
- PIC17Cxx: de topklasse



Figuur 7/6.5.1-1: Evolutie van de PIC16/17-familie.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.1-2: Overzicht van de PIC16/17 families.

Alle families zijn verkrijgbaar in verschillende behuizingen en bieden One-Time-Programmable, laagspanning en laagvermogen opties. Van sommige typen zijn ROM en herprogrammeerbare typen beschikbaar.

De OTP-optie biedt de gebruiker veel voordelen voor een prijs die overeenkomt met die van concurrerende ROM-oplossingen, zoals korte time-to-market, gemakkelijk codes te wijzigen, minder voorraad nodig en mogelijkheid om de oplossing naderhand aan te passen aan de eisen van de gebruiker.

PIC16C5x: basisklasse

De PIC16C5x-familie is de meest gebruikte en goedkoopste groep. Deze typen hebben een 12 bit brede instructieset en zijn op dit moment verkrijgbaar in 18-, 20- of 28-pens behuizingen (ook SOIC en SSOP). Deze familie heeft ook typen die met een voedingsspanning van minimaal 2 V kunnen werken en daardoor zeer geschikt zijn voor draagbare (batterij gevoede) apparatuur.

	Clock	Memory			Peripherals		Features		
	Maximum Frequency of Operation (MHz)	Program Memory (words) EPROM	Program Memory (words) ROM	RAM Data Memory (bytes)	Timer Module(s)	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages
PIC16C54	20	512	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16C54A	20	512	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16CR54	20	—	512	25	TMR0	12	2.0-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16C55	20	512	—	25	TMR0	20	2.5-6.25	33	28-pin DIP, 28-pin SOIC, 28 pin SSOP
PIC16C56	20	1K	—	25	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C57	20	2K	—	72	TMR0	20	2.5-6.25	33	28-pin DIP, 28-pin SOIC, 28 pin SSOP
PIC16CR57A	20	—	2K	72	TMR0	20	2.0-6.25	33	28-pin DIP, 28-pin SOIC, 28 pin SSOP
PIC16C58A	20	2K	—	73	TMR0	12	2.5-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP
PIC16CR58A	20	—	2K	73	TMR0	12	2.0-6.25	33	18-pin DIP, 18-pin SOIC, 20 pin SSOP

Note: All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

Figuur 7/6.5.1-3: Overzicht van de PIC16C5x en enhanced (A) PIC16C5x-typen.

6.5 PIC16/17 typen 8 bit microcontrollers

	Clock	Memory				Peripherals								Features			
	Maximum Frequency of Operation (MHz)	Program Memory: EPROM	Program Memory: EEPROM	Data Memory (bytes)	Data EEPROM (bytes)	Timer Module(s)	Capture/Compare/PWM Module(s)	Serial Port(s) (SPI/I²C, SCI)	Parallel Slave Port	Analog to Digital Converter (8-bit)	Comparator(s)	Internal Reference Voltage	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Brown-out	Packages
PIC16C61	20	1K	—	36	—	TMR0	—	—	—	—	—	—	3	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C62*	20	2K	—	128	—	TMR0, TMR1, TMR2	2	SPI/I²C	—	—	—	—	10	22	2.5-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C63*	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	—	—	—	—	10	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C64	20	2K	—	128	—	TMR0, TMR1, TMR2	1	SPI/I²C	Yes	—	—	—	8	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C65	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	Yes	—	—	—	11	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C620*	20	512	—	80	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C621*	20	1K	—	80	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C622	20	2K	—	128	—	TMR0	—	—	—	—	2	Yes	4	13	3.0-6.0	Yes	18-pin DIP, 18-pin SOIC, 20-pin SSOP
PIC16C71	20	1K	—	36	—	TMR0	—	—	—	4 ch	—	—	4	13	3.0-6.0	—	18-pin DIP, 18-pin SOIC
PIC16C73	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	—	5 ch	—	—	11	22	3.0-6.0	—	28-pin SDIP, 28-pin SOIC
PIC16C74	20	4K	—	192	—	TMR0, TMR1, TMR2	2	SPI/I²C/SCI	Yes	8 ch	—	—	12	33	3.0-6.0	—	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC16C84	10	—	1K	36	64	TMR0	—	—	—	—	—	—	4	13	2.0-6.0	—	18-pin DIP, 18-pin SOIC

* *Please contact your local sales office for availability of these devices.

Note 1: All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

2: The PIC16CXX Timer1 has its own oscillator circuit and can operate asynchronously to the device. Timer1 can increment while the device is in SLEEP mode. This allows a Real Time Clock to be implemented.

3: PORTB has software-configurable weak pull-ups.

Figuur 7/6.5.1-4: Overzicht van de PIC16Cxx-typen.

PIC16Cxx: middenklasse

De microcontrollers uit de PIC16Cxx-familie hebben een groot aantal optionele eigenschappen, 18- tot 44-pens behuizingen en integratie met de periferie op zowel laag als hoog niveau. Deze familie heeft een 14 bit brede instructieset, de mogelijkheid tot het behandelen van interrupts en een diepere (acht niveaus) hardware stack. De PIC16Cxx-familie is daardoor bestemd voor ingewikkelder toepassingen dan de basisgroep.

PIC17Cxx: topklasse

De PIC17Cxx is een "high-performance" familie die sneller werkt dan alle, in de industrie verkrijgbare 8 bit microcontrollers. De RISC-architectuur van deze typen heeft 16 bit brede instructiewoorden waardoor een uitgebreider instructieset en krachtige, op vectoren gebaseerde interrupt-handling zijn ontstaan. De chip's zijn voorzien van zeer bruikbare voorzieningen, zoals timers, ingebedde A/D-omzetters, groter instructie/datageheugen, communicatie-schakelingen (I²C-bus, SPI en USART's) en ROM-, RAM-, EPROM- en EEPROM-geheugens. Deze familie wordt voortdurend aangevuld met nieuwe typen.

Zowel de PIC16Cxx als de PIC17Cxx familie worden ondersteund met zeer toegankelijke ontwikkelsystemen (onder andere PICstart), inclusief assembler, software simulator, C-compiler, fuzzy-logic ontwikkelings-software, programmeer-apparatuur en in-circuit emulators (figuur 7/6.5.1-6).

6.5 PIC16/17 typen 8 bit microcontrollers

	Clock	Memory		Peripherals				Features					
	Maximum Frequency of Operation (MHz)	Program Memory EPROM	RAM Data Memory (bytes)	Timer Module(s)	Captures	PWMs	Serial Port(s) (SCI)	External Interrupts	Interrupt Sources	I/O Pins	Voltage Range (Volts)	Number of Instructions	Packages
PIC17C42	25	2K	232	TMR0,TMR1, TMR2,TMR3	2	2	Yes	Yes	11	33	4.5-5.5	55	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC17C43*	25	4K	454	TMR0,TMR1, TMR2,TMR3	2	2	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP, 44-pin PLCC, 44-pin QFP
PIC17C44	25	8K	454	TMR0,TMR1, TMR2,TMR3	2	2	Yes	Yes	11	33	2.5-6.0	58	40-pin DIP, 44-pin PLCC, 44-pin QFP

* Please contact your local sales office for availability of these devices.

Note 1: All PIC16/17 Family devices have Power-On Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

2: The PIC17C4X devices can also operate in microprocessor and external microcontroller modes.

3: PORTB has software-configurable weak pull-ups.

Figuur 7/6.5.1-5: Overzicht van de PIC17Cxx-typen.

Development Tool	Name	PIC16C5X	PIC16CXX	PIC17CXX
Assembler	MPASM	✔	✔	✔
Software Simulator	MPSIM	✔	✔	✔
C Compiler*	MP-C	✔	✔	✔
Entry Level Development Kit	PICSTART®	✔	✔	Planned
Universal Programmer	PRO MATE™	✔	✔	✔
Universal In-Circuit Emulator	PICMASTER®	✔	✔	✔
Fuzzy Logic Development Tool	fuzzyTECH®-MP	✔	✔	✔

Figuur 7/6.5.1-6: Synergische ontwikkel-gereedschappen voor de PIC16/17 families.

Afspraken betreffende de benamingen
Aangezien de PIC16/17 architectuur zoveel mogelijkheden voor allerlei opties biedt, is het noodzakelijk van te voren afspraken te maken over de benamingen van de diverse typen. In figuur 7/6.5.1-7 is een overzicht hiervan te zien. Hierbij geldt dat de nummering binnen elke familie niet echt iets hoeft te betekenen en dat van sommige typen de maximale frequentie lager kan zijn dan 20 MHz.

6.5 PIC16/17 typen 8 bit microcontrollers

	Family	Architectural Features	Name	Technology	Products
PIC16C5X	Base-Line 8-bit Microcontroller Family	• 12-bit wide instruction set • DC - 20 MHz clock speed • 200 ns instruction cycle	PIC16C5X PIC16C5XA (Note 1)	OTP program memory, digital only	PIC16C54 PIC16C54A PIC16C55 PIC16C56 PIC16C57 PIC16C58A
			PIC16CR5X PIC16CR5XA (Note 1)	ROM program memory, digital only	PIC16CR54 PIC16CR57A PIC16CR58A
PIC16CXX	Mid-Range 8-bit Microcontroller Family	• 14-bit wide instruction set • Internal/external interrupts • DC - 20 MHz clock speed (Note 3) • 200 ns instruction cycle (@ 20 MHz)	PIC16C6X	OTP program memory, digital	PIC16C61 PIC16C62 PIC16C63 PIC16C64 PIC16C65
			PIC16CR6X	ROM program memory, digital only	Planned
			PIC16C62X	OTP program memory with comparators	PIC16C620 PIC16C621 PIC16C622
			PIC16C7X	OTP program memory, with analog functions (e.g. A/D)	PIC16C71 PIC16C73 PIC16C74
			PIC16C8X	EEPROM program and data memory	PIC16C84
			PIC16CR8X	ROM program and EEPROM data memory	Planned
PIC17CXX	High-End 8-bit Microcontroller Family	• 16-bit wide instruction set • Internal/external interrupts • DC - 25 MHz clock speed • 160 ns instruction cycle	PIC17C4X	OTP program memory, digital only	PIC17C42 PIC17C43 PIC17C44
			PIC17CR4X	ROM program memory, digital only	Planned

Figuur 7/6.5.1-7: Vastgestelde PIC16/17 benamingen.
Note 1: de "A" duidt op een geavanceerder proces-technologie (meestal geringere dissipatie, hogere snelheid, enzovoorts).

Pin-compatibiliteit

Schakelingen die dezelfde soort behuizing hebben, terwijl V_{DD}, V_{SS} en $\overline{MCLR}$ zich op dezelfde plaatsen bevinden, worden pin-compatibel genoemd. Deze verschillende schakelingen kunnen dan in hetzelfde voetje worden gestoken, maar hebben waarschijnlijk een iets andere software nodig om op de juiste manier te werken. Let op dat niet alle schakelingen met dezelfde behuizing ook werkelijk pin-compatibel zijn. De PIC16C62 is bijvoorbeeld wel compatibel met de PIC16C63, maar niet met de PIC16C55.

Pin Compatible Devices	Package
PIC16C61, PIC16C620, PIC16C621, PIC16C622, PIC16C71, PIC16C84, PIC16C54, PIC16C54A, PIC16CR54, PIC16C56, PIC16C58A, PIC16CR58A	18 pin
PIC16C62, PIC16C63, PIC16C73	28 pin
PIC16C55, PIC16C57, PIC16CR57A	28 pin
PIC17C42, PIC17C43, PIC17C44	40 pin
PIC16C64, PIC16C65, PIC16C74	40 pin

Figuur 7/6.5.1-8: Overzicht van pin-compatibele schakelingen.

6.5 PIC16/17 typen 8 bit microcontrollers

7/6.5.2

Type-beschrijving van de PIC16C5x-typen

Inleiding

Statische CMOS microcontrollers

De PIC16C5x is een familie goedkope, 8 bit, volledig statische, op EPROM gebaseerde CMOS microcontrollers.
Deze familie is pin- en software-compatibel met de Enhanced PIC16C5x-familie.
Er wordt gebruik gemaakt van een RISC-achtige architectuur met slechts 33 single word/single cycle instructies. Alle instructies worden uitgevoerd in één cyclus (200 ns), behalve branch-instructies die twee cycli nodig hebben. De 12 bit brede instructies zijn in hoge mate symmetrisch, hetgeen resulteert in een 2:1 code compressie ten opzichte van andere 8 bit microcontrollers.
De PIC16C5x produkten zijn voorzien van speciale kenmerken om de systeemkosten en de eisen die aan de voeding worden gesteld te beperken. De Power-On Reset (POR) en de Device Reset Timer maken externe reset-schakelingen overbodig. Er kan uit vier oscillator-configuraties worden gekozen, inclusief de LP (Low Power) en de goedkope RC-oscillator. Verder kan gebruik worden gemaakt van de SLEEP mode, de watchdog timer en code beveiligingen.
De UV-wisbare CERDIP-versies zijn ideaal voor het ontwikkelen van code, terwijl de goedkopere OTP-versies bruikbaar zijn voor de produktie (bij alle aantallen).
De PIC16C5x produkten worden ondersteund door een volledig toegeruste macro-assembler, een software simulator, een in-circuit emulator, een C-compiler, fuzzy-logic tools, een goedkope ontwikkel-programmer en een complete programmer. Alle tools zijn bruikbaar op IBM PC's en vergelijkbare typen.

Algemene gegevens

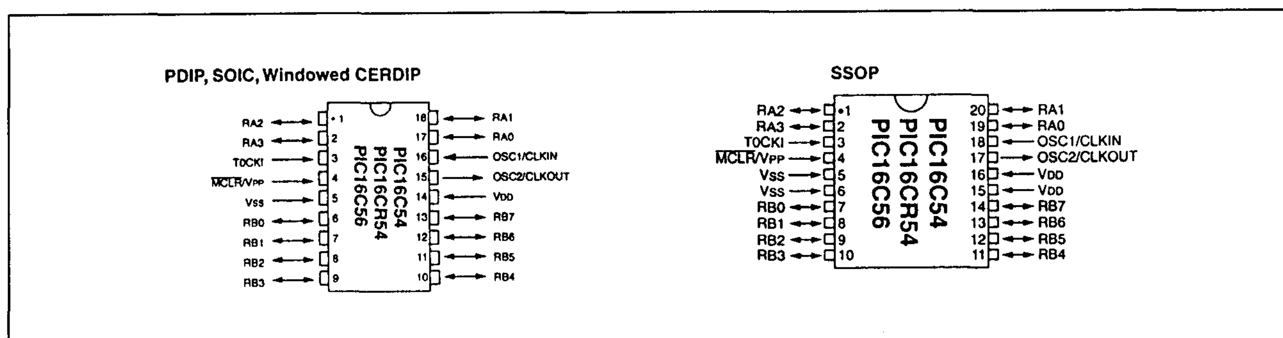
- behandelde typen: PIC16C54, PIC16CR54, PIC16C55, PIC16C56, PIC16C57 (zie ook figuur 7/6.5.2-1)
- 33 single-word instructies
- single-cycle instructies (200 ns) behalve branches
- snelheid:
  DC - 20 MHz clock input
  DC - 200 ns instructie-cyclus
- 12 bit brede instructies
- 8 bit breed data-pad
- 7 of 8 special function hardware registers
- twee niveaus diepe hardware stack
- directe, indirecte en relatieve adresseermodes voor data en instructies
- 8 bit real-time clock/counter (TMR0) met 8 bit programmeerbare prescaler
- power-on reset (POR) en device reset timer
- watchdog timer (WDT) met eigen on-chip RC-oscillator
- programmeerbare code-beveiliging
- energie besparende SLEEP mode
- EPROM/ROM selecteerbare oscillator-opties:
  - RC: low cost RC-oscillator
  - XT: standaard kristal/resonator
  - HS: high speed kristal/resonator
  - LP: energie besparend laagfrequent kristal
- CMOS: volledig statisch ontwerp

6.5 PIC16/17 typen 8 bit microcontrollers

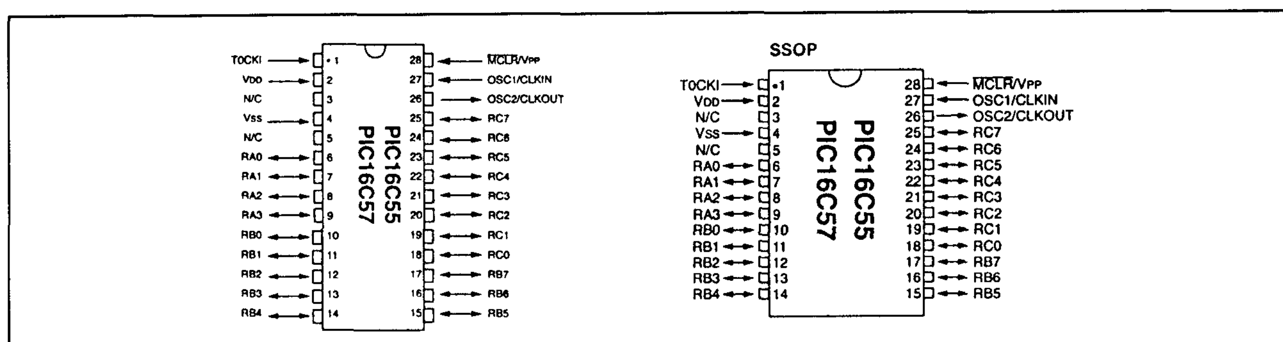
- voedingsspanningen:
  EPROM-typen: 2,5 V - 6,25 V
  ROM-type: 2 V - 6,25 V
  EPROM/ROM (automotive): 2,5 V - 6 V
- laag energie-verbruik:
  <2 mA typ. (5 V, 4 MHz)
  15 µA typ. (3V, 32 kHz)
  <3 µA typ. (standby)
- fabrikant: Microchip Technology

Device	Pins	I/O	EPROM/ ROM	RAM
PIC16C54	18	12	512	25
PIC16CR54	18	12	512	25
PIC16C56	18	12	1K	25
PIC16C55	28	20	512	24
PIC16C57	28	20	2K	72

Figuur 7/6.5.2-1: Overzicht van de PIC16C5x-typen.

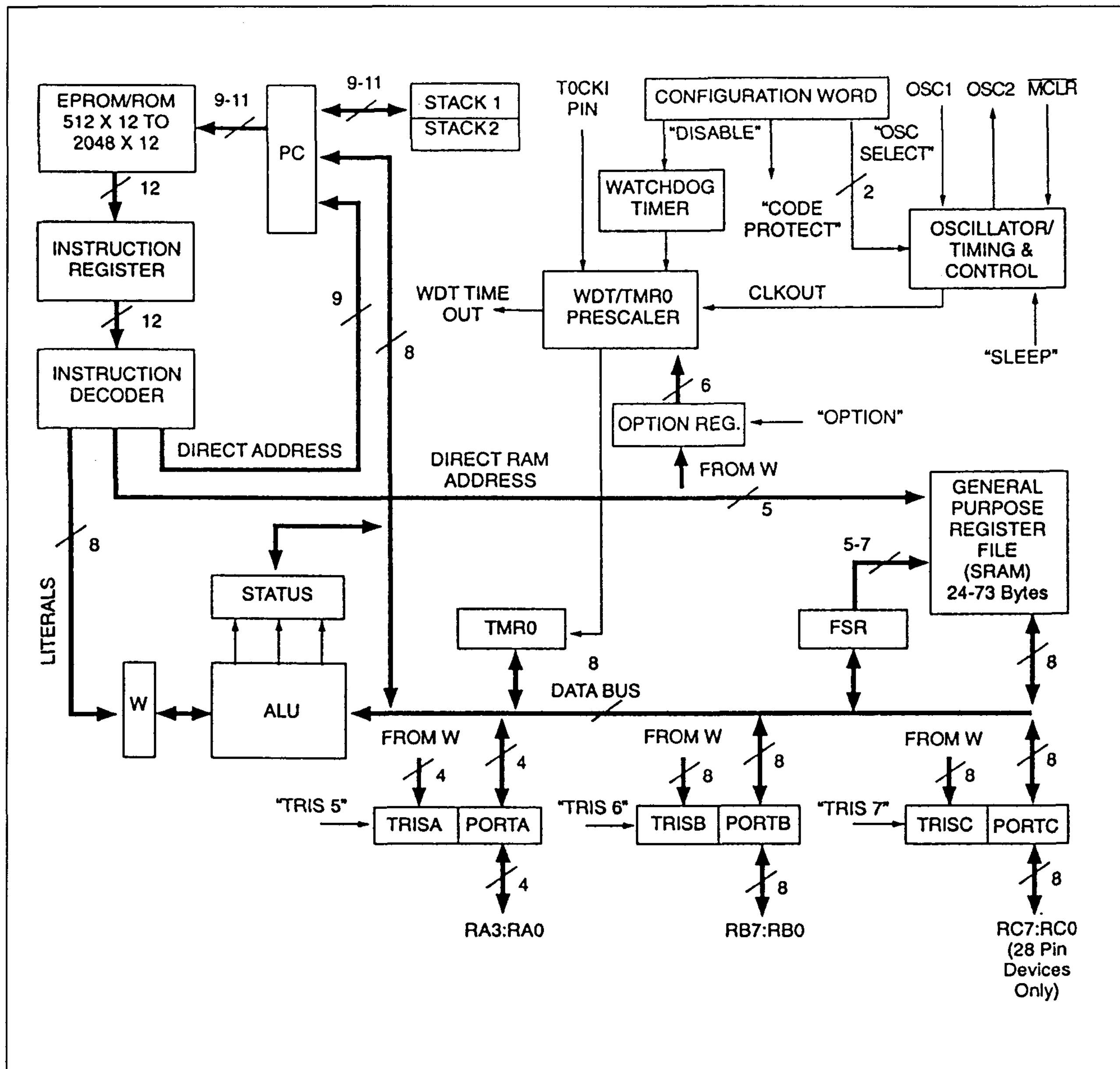


Figuur 7/6.5.2-2: Aansluitingen van de PIC16C54/CR54/C56-typen, links in PDIP, SOIC en CERDIP uitvoering (met venster), rechts SSOP.



Figuur 7/6.5.2-3: Aansluitingen van de PIC16C55/C57-typen, links in PDIP, SOIC en CERDIP uitvoering (met venster), rechts SSOP.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-4: Vereenvoudigd blokschema van de PIC16C5x-schakelingen.

Architectuur

Inleiding

Prestaties als van de PIC16C5x-familie komen normaal alleen voor bij RISC-microprocessoren. Om te beginnen gebruikt de PIC16C5x een Harvard architectuur waarbij toegang tot programma en data via aparte bussen plaats vindt. Hierdoor wordt de bandbreedte verbeterd ten opzichte van de traditionele von Neumann architectuur (waarbij programma en data via dezelfde bus worden bereikt). Door programma- en datageheugen van elkaar te scheiden kunnen instructies ook andere afmetingen hebben dan het 8 bit brede datawoord.

6.5 PIC16/17 typen 8 bit microcontrollers

Name	DIP, SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
RA0	17	19	I/O	TTL	Bi-directional I/O port
RA1	18	20	I/O	TTL	
RA2	1	1	I/O	TTL	
RA3	2	2	I/O	TTL	
RB0	6	7	I/O	TTL	Bi-directional I/O port
RB1	7	8	I/O	TTL	
RB2	8	9	I/O	TTL	
RB3	9	10	I/O	TTL	
RB4	10	11	I/O	TTL	
RB5	11	12	I/O	TTL	
RB6	12	13	I/O	TTL	
RB7	13	14	I/O	TTL	
T0CKI	3	3	I	ST	Clock input to TMR0 timer. Must be tied to Vss or VDD, if not in use, to reduce current consumption.
$\overline{\text{MCLR}}$/VPP	4	4	I	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device. Voltage on $\overline{\text{MCLR}}$/VPP must not exceed VDD to avoid unintended entering of test modes.
OSC1/CLKIN	16	18	I	ST	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	17	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
VDD	14	15,16	P	—	Positive supply for logic and I/O pins.
Vss	5	5,6	P	—	Ground reference for logic and I/O pins.

Legend: I=input, O=output, I/O=input/output, P=power, — =Not Used, TTL=TTL input, ST=Schmitt Trigger input

Figuur 7/6.5.2-5: Aansluitingen en signaalfuncties van de PIC16C54, PIC16CR54 en PIC16C56.

Met een 12 bit brede programma-bus kan een 12 bit instructie in één enkele cyclus worden opgehaald.
Met een tweetraps pijplijn overlappen fetch en executie van instructies elkaar. Als gevolg daarvan worden alle instructies in één enkele cyclus (200 ns bij 20 MHz) uitgevoerd, met uitzondering van program-branches. De PIC16C58 en PIC16C57 adresseren programma-geheugen ter grootte van 2 kB x 12, de PIC16C55 en PIC16C56 adresseren 1 kB x 12 en de PIC16C54 adresseert 512 B x 12. Al het programma-geheugen is intern. De PIC16C5x kan zijn register-files en data-geheugen direct of indirect adresseren. Alle registers met speciale functies (ook de program-counter) zijn in het data-geheugen opgenomen.

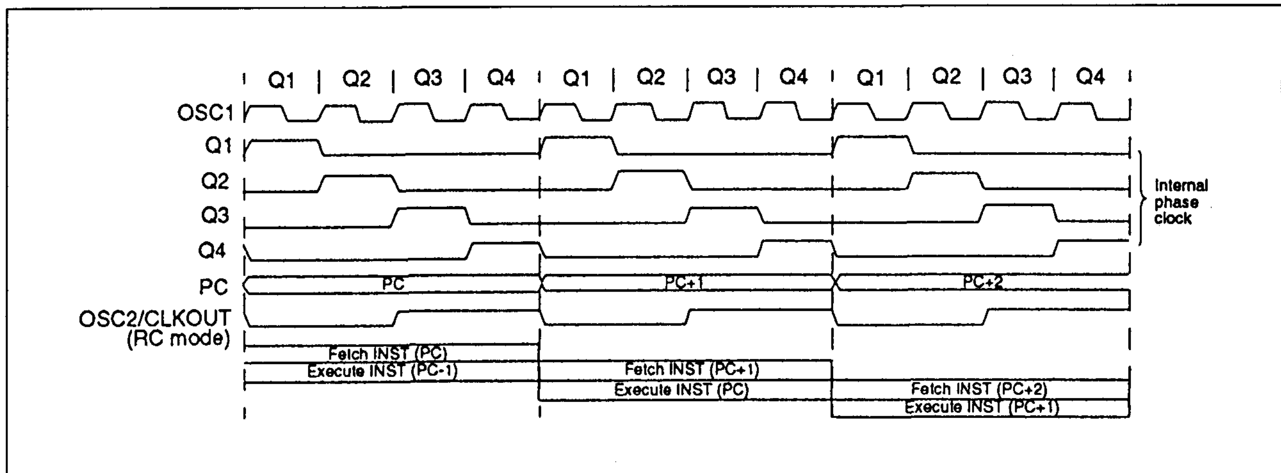
6.5 PIC16/17 typen 8 bit microcontrollers

Name	DIP, SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
RA0	6	5	I/O	TTL	Bi-directional I/O port
RA1	7	6	I/O	TTL	
RA2	8	7	I/O	TTL	
RA3	9	8	I/O	TTL	
RB0	10	9	I/O	TTL	Bi-directional I/O port
RB1	11	10	I/O	TTL	
RB2	12	11	I/O	TTL	
RB3	13	12	I/O	TTL	
RB4	14	13	I/O	TTL	
RB5	15	15	I/O	TTL	
RB6	16	16	I/O	TTL	
RB7	17	17	I/O	TTL	
RC0	18	18	I/O	TTL	Bi-directional I/O port
RC1	19	19	I/O	TTL	
RC2	20	20	I/O	TTL	
RC3	21	21	I/O	TTL	
RC4	22	22	I/O	TTL	
RC5	23	23	I/O	TTL	
RC6	24	24	I/O	TTL	
RC7	25	25	I/O	TTL	
TOCKI	1	2	I	ST	Clock input to TMR0 register. Must be tied to V_{SS} or V_{DD}, if not in use, to reduce current consumption.
$\overline{MCLR}$/V_{PP}	28	28	I	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device. Voltage on $\overline{MCLR}$/V_{PP} must not exceed V_{DD} to avoid unintended entering of test modes.
OSC1/CLKIN	27	27	I	ST	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	26	26	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
V_{DD}	2	3,4	P	—	Positive supply for logic and I/O pins.
V_{SS}	4	1,14	P	—	Ground reference for logic and I/O pins.
N/C	3,5	—	—	—	Unused, do not connect

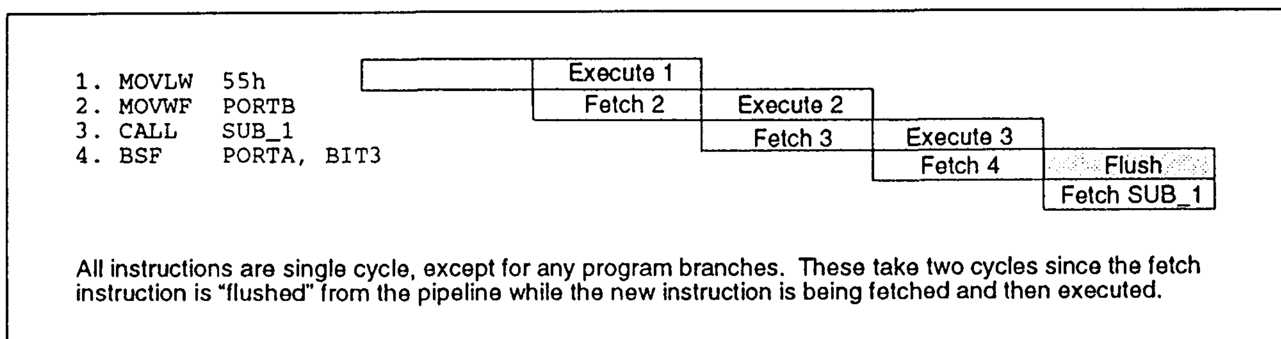
Legend: I=input, O=output, I/O=input/output, P=power, — =Not Used, TTL=TTL input, ST=Schmitt Trigger input

Figuur 7/6.5.2-6: Aansluitingen en signaalfuncties van de PIC16C55 en PIC16C57.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-7: Clock/instructie-cyclus.



Figuur 7/6.5.2-8: Voorbeeld van de voortgang van instructies in de pijplijn.

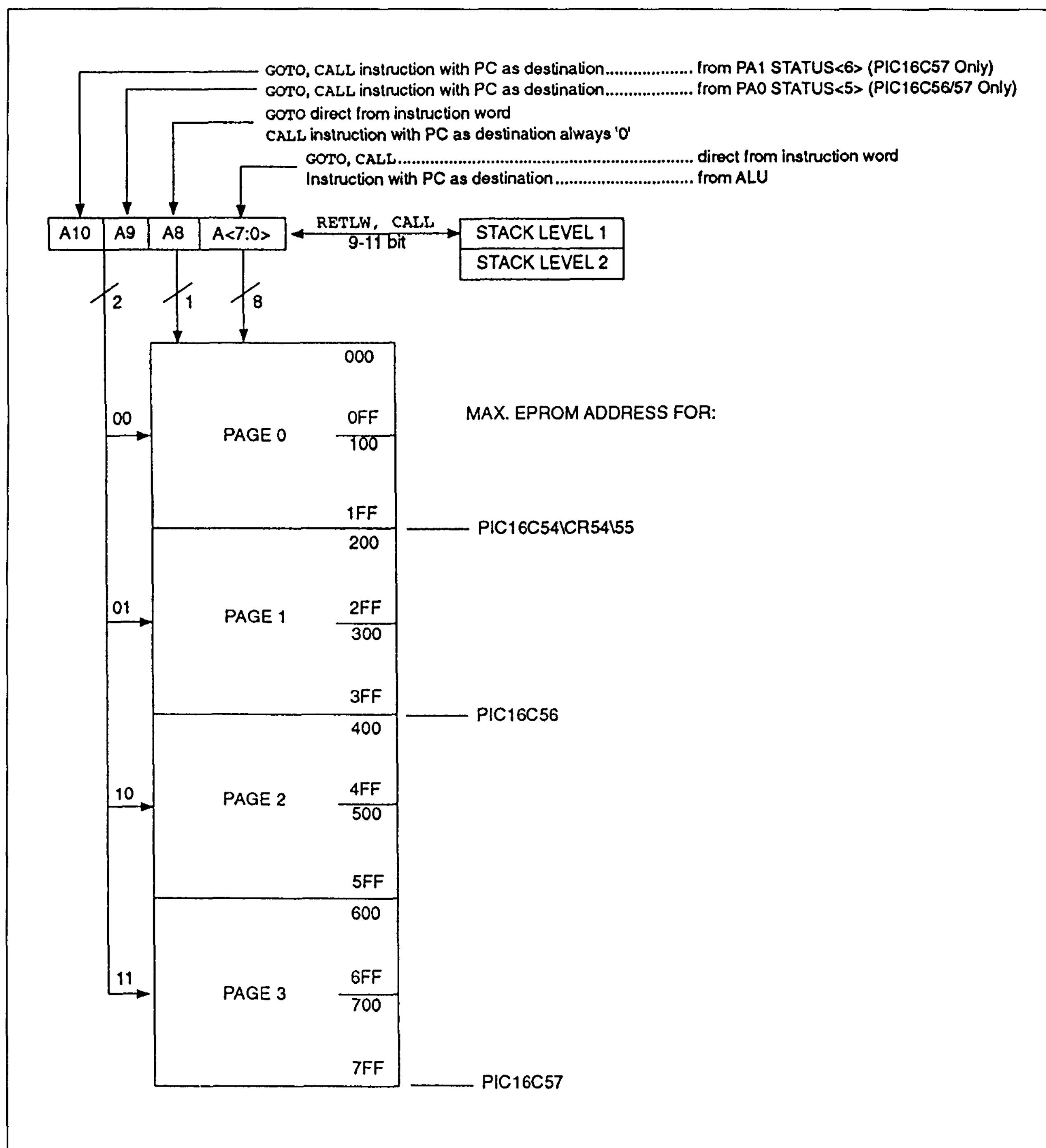
De PIC16C5x heeft een uiterst orthogonale (symmetrische) instructieset, waardoor het mogelijk is elke willekeurige operatie, gebruik makend van welke adresseringsmode dan ook, op elk willekeurig register uit te voeren. Een PIC16C5x-schakeling bevat een 8 bit ALU en werk-register. De ALU is een algemene rekenkundige eenheid waarmee rekenkundige en Boole'se functies tussen data in het werkregister en een willekeurige register-file worden uitgevoerd (optellen, aftrekken, schuif- en logische operaties). Tenzij anders vermeld zijn rekenkundige operaties two's complement van aard. Bij 2-operand instructies is gewoonlijk één operand het werkregister en de andere een file-register of een immediate constante. Bij 1-operand instructies is de operand het W-register of een file-register. Het W-register is een 8 bit werkregister dat voor ALU-operaties wordt gebruikt. Het is niet adresseerbaar. Afhankelijk van de instructie die wordt uitgevoerd kan de ALU de waarden van de Carry (C), Digit Carry (DC) en Zero (Z) bits in het statusregister beïnvloeden. De C en DC bits werken bij aftrekken respectievelijk als borrow en digit borrow out bit. (Zie bijvoorbeeld de SUBWF en ADDWF instructies).

Clock-schema/Instructie-cyclus

Het clock-ingangssignaal (OSC1/CLKIN) wordt inwendig door vier gedeeld om vier niet-overlappende kwadratuur clocksignalen (Q1, Q2, Q3 en Q4) te genereren. Intern wordt de program-counter (PC) elke Q1 met één verhoogd, terwijl de instructie uit het programma-geheugen wordt opgehaald en op Q4 wordt opgeborgen in het instruc-

6.5 PIC16/17 typen 8 bit microcontrollers

tie-register. De instructie wordt gedurende de volgende Q1 tot en met Q4 gedecodeerd en uitgevoerd. In figuur 7/6.5.2-7 en het voorbeeld in figuur 7/6.5.2-8 is het verband tussen de clock-signalen en de instructies te zien.



Figuur 7/6.5.2-9: Organisatie van het programma-geheugen.

Instructie-afhandeling/pijplijnen

Een instructie-cyclus bestaat uit vier Q-cycli. Het ophalen van de instructie en het uitvoeren daarvan worden gepijplijnd, zodat het ophalen één instructie-cyclus duurt, terwijl voor het decoderen en uitvoeren nog een instructie-cyclus nodig is.
Effectief duurt door het pijplijnen elke instructie slechts één cyclus. Als door een instructie de program-counter verandert (bijvoorbeeld door GOTO) zijn twee cycli nodig om de instructie uit te voeren (zie figuur 7/6.5.2-8).
Een ophaal-cyclus (fetch) begint met het ophogen van de program-counter bij Q1. In de executie-cyclus wordt de opgehaalde instructie bij Q1 in het Instructie Register (IR) gelatched.
Deze instructie wordt daarna gedurende Q2, Q3 en Q4 gedecodeerd en uitgevoerd. Het data-geheugen wordt bij Q2 gelezen (operand read) en weggeschreven bij Q4 (destination write).

Geheugen-organisatie

Programma-geheugen

Er kunnen maximaal 512 woorden met een breedte van 12 bit direct in het on-chip programma-geheugen (EPROM of ROM) worden geadresseerd. Grotere programmageheugens kunnen worden geadresseerd door één van de vier beschikbare pagina's van 512 woorden per stuk te selecteren (zie figuur 7/6.5.2-9).
Het na elkaar volgen van de instructies wordt geregeld via de Program Counter (PC) die automatisch met één wordt verhoogd voor het uitvoeren van in-line programma's. Programma-besturingsoperaties die directe, indirecte en relatieve adresseringsmodes ondersteunen kunnen worden uitgevoerd met bit-test, skip, call of jump-instructies of door het laden van berekende adressen in de PC.
Bovendien wordt een on-chip stack van twee niveaus toegepast voor gemakkelijk te gebruiken subroutine-nesting.

Data-geheugen

De 8 bit databus verbindt twee functionele basiselementen met elkaar: de Register File (samengesteld uit maximaal 80 adresseerbare 8 bit registers, inclusief de I/O-poorten) en de 8 bit brede Arithmetic Logic Unit.
32 bytes RAM zijn direct adresseerbaar, terwijl een "banking" stelsel (met banken van 16 bytes per stuk) wordt gebruikt om grotere data-geheugens te adresseren (zie figuur 7/6.5.2-10). Data kan óf direct worden geadresseerd óf indirect met gebruikmaking van het file-select register. Immediate adressering wordt ondersteund door speciale "letterlijke" instructies die data laden vanuit programma-geheugen naar het W-register.
De register file is opgedeeld in twee functionele groepen: Special Function registers en General Purpose registers. Tot de Special Function registers behoren het Timer0-register (TMR0), de Program Counter (PC), het Status Register, de I/O-registers (poorten) en het File Select Register (FSR). De algemene registers (general purpose) worden gebruikt voor data en besturingsinformatie onder commando van de instructies.
Bovendien worden special purpose registers gebruikt om de I/O-poort configuratie en prescaler opties te regelen.

General Purpose Register File

De register file is direct toegankelijk of indirect via het file select register (zie indirecte data adressering).

Special Function Registers

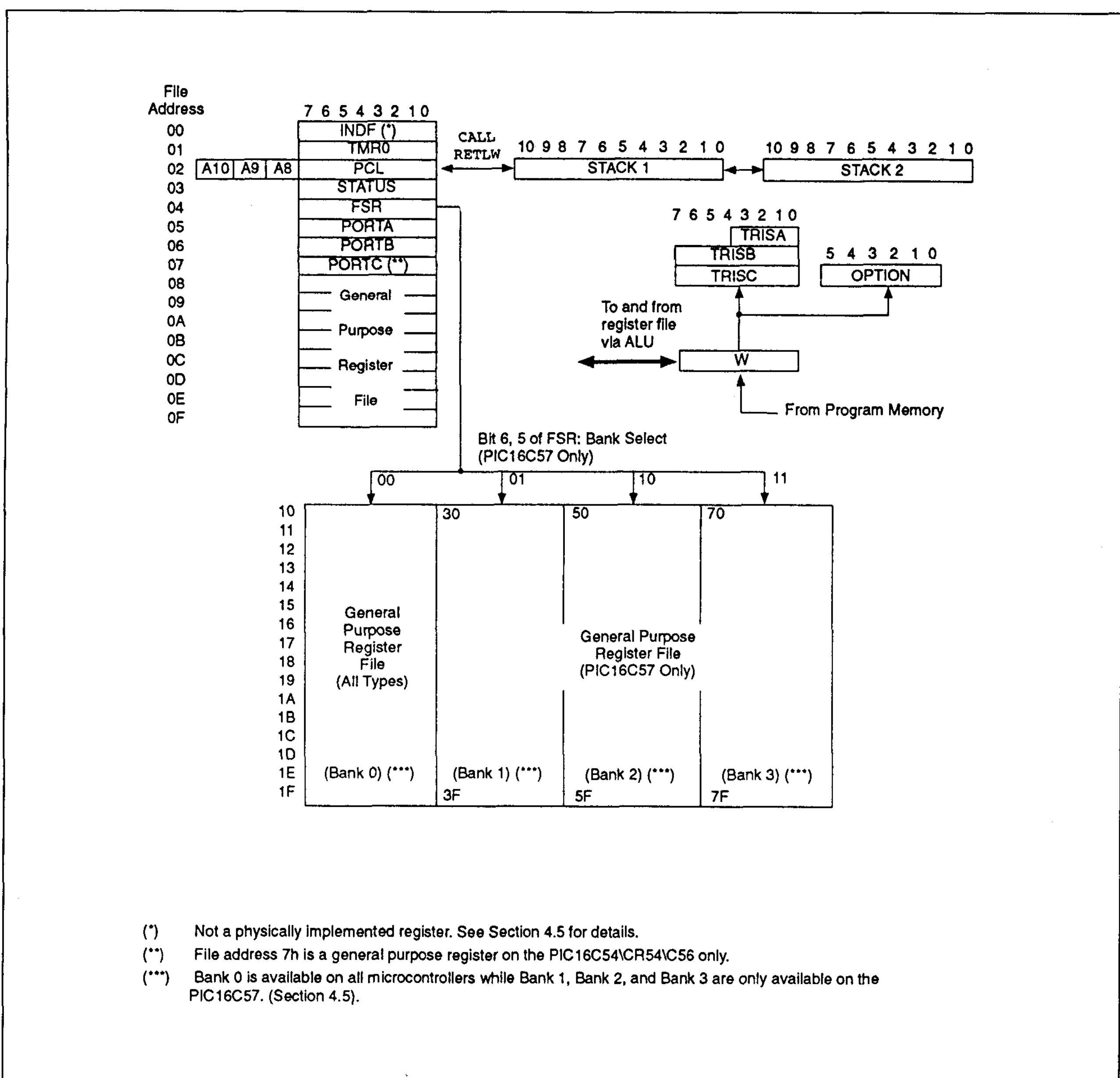
De registers voor de speciale functies worden door de CPU en perifere functies gebruikt om de werking van de schakeling te regelen (zie tabel 7/6.5.2-1).

Status Register

Dit register bevat de rekenkundige status van de ALU, de RESET status en de paginapreselect bits voor programma-geheugens die groter zijn dan 512 woorden (zie figuur 7/6.5.2-11). Net als alle andere registers kan het statusregister de bestemming zijn van

6.5 PIC16/17 typen 8 bit microcontrollers

alle instructies. De STATUS bits worden echter pas gezet na de volgende schrijfoperatie. Bovendien zijn de $\overline{TO}$ en $\overline{PD}$ bits niet beschrijfbaar. Daardoor kan het resultaat van een instructie met het statusregister anders zijn dan de bedoeling was.
CLRF STATUS zal bijvoorbeeld alle bits leegmaken behalve $\overline{TO}$ en $\overline{PD}$, waarna het Z-bit wordt gezet en het statusregister achterblijft met: 000u u100 (u = onveranderd). Daarom is het beter alleen BCF, BSF en MOVWF instructies te gebruiken om het statusregister te veranderen, omdat deze instructies geen enkel status-bit beïnvloeden (zie ook de instructieset).



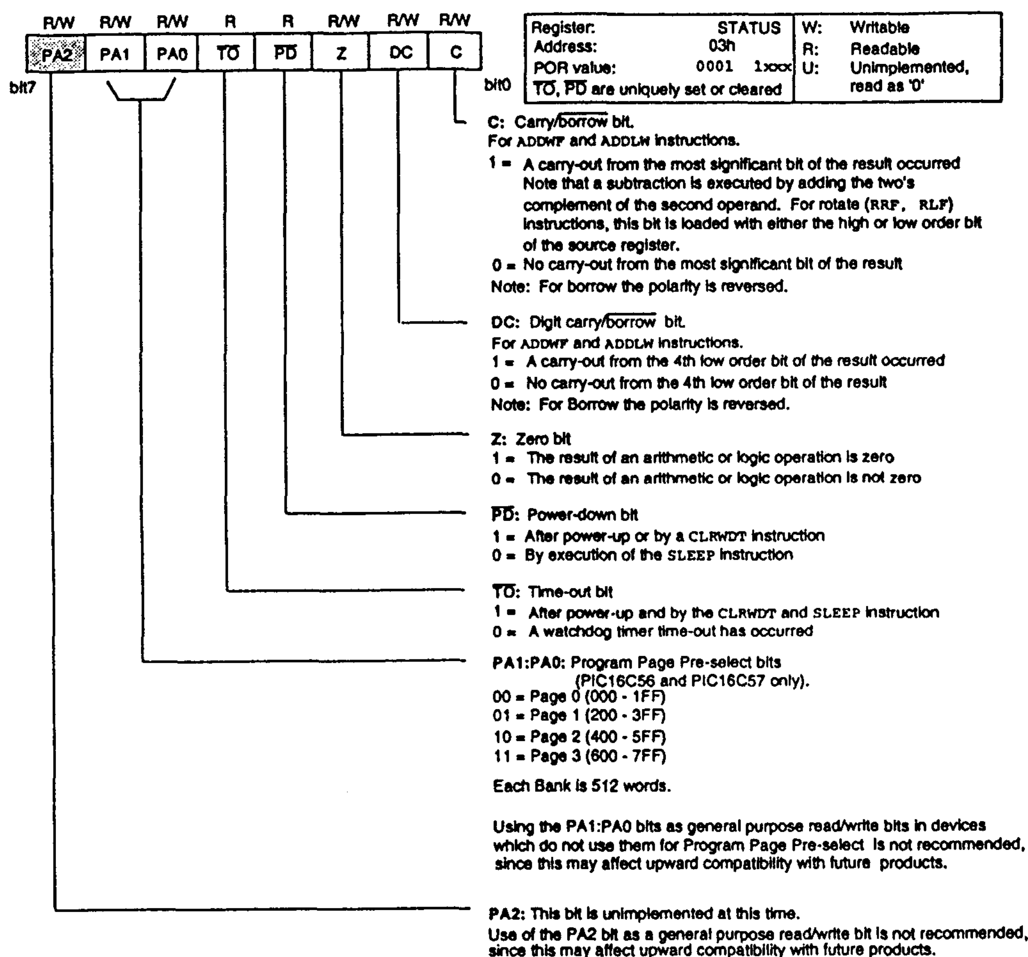
Figuur 7/6.5.2-10: Inrichting van het data-geheugen.

6.5 PIC16/17 typen 8 bit microcontrollers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on MCLR and WDT Reset
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---- ----	---- ----
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of PC								1111 1111	1111 1111
03h	STATUS	PA2	PA1	PA0	TO	PD	Z	DC	C	0001 1xxx	000? ?uuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA	—	—	—	—	RA3	RA2	RA1	RA0	---- xxxx	---- uuuu
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
07h	PORTC (Note 2)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu

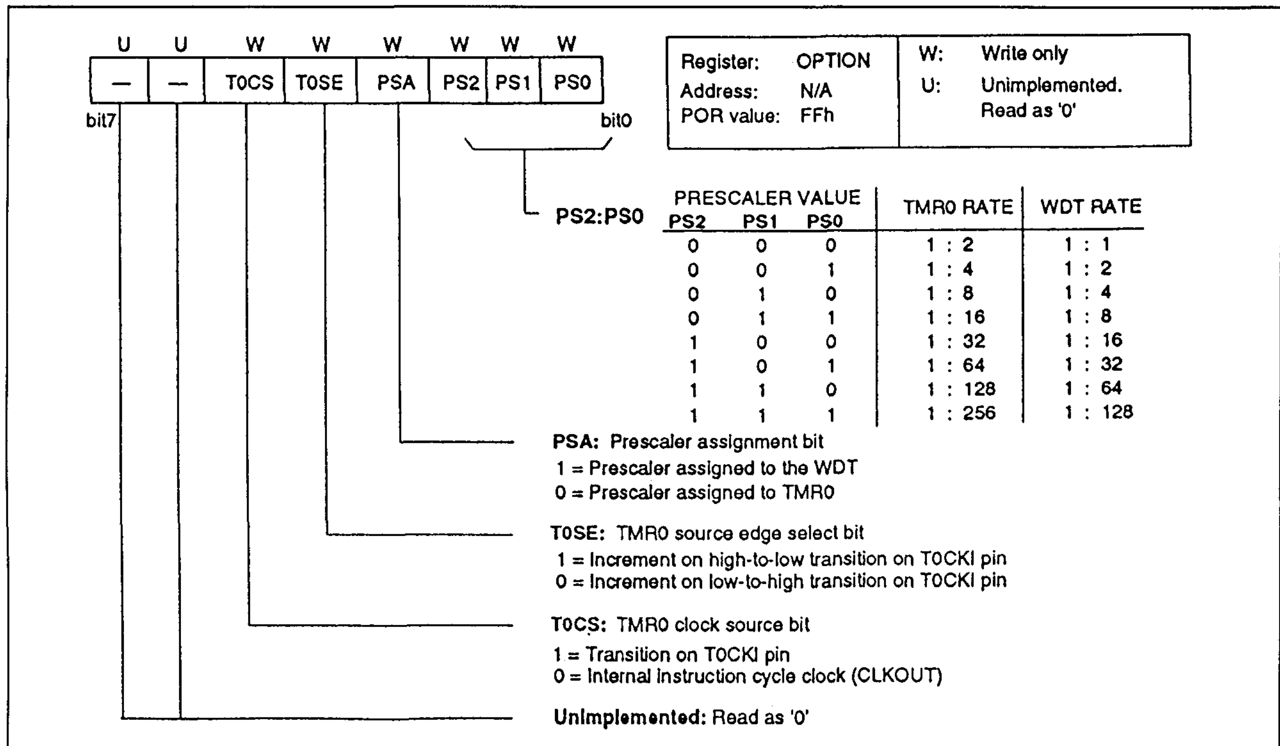
Legend: x = unknown, u = unchanged. - = Unimplemented, Read as '0'.
Note 1: The upper byte of the program counter is not directly accessible. The upper bits can be set or cleared by writing to PA1:PA0 (STATUS<6:5>).
2: File address 7h is a general purpose register on the PIC16C54\CR54\C56.
3: Shading indicates unimplemented bits.

Tabel 7/6.5.2-1: Overzicht van de PIC16C5x register-files.



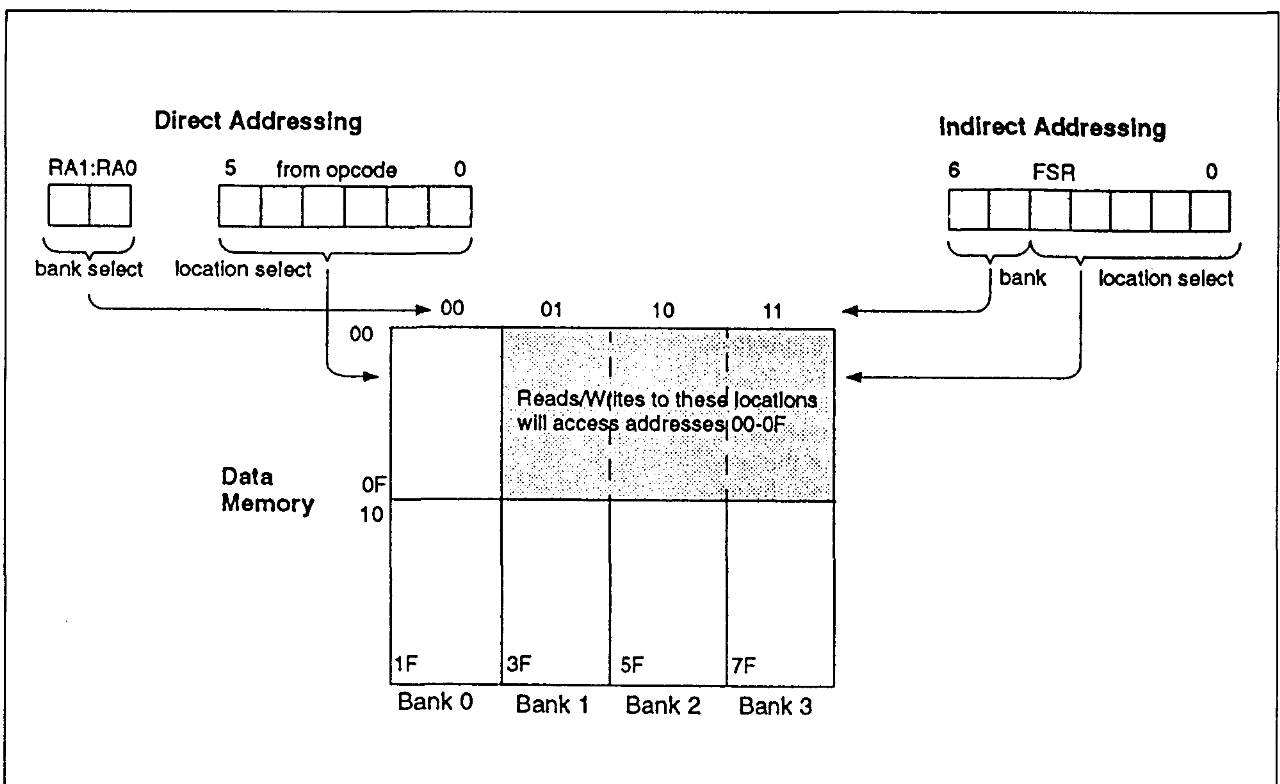
Figuur 7/6.5.2-11: Betekenis van de bits in het statusregister.

6.5 PIC16/17 typen 8 bit microcontrollers



PS2	PS1	PS0	TMR0 RATE	WDT RATE
0	0	0	1 : 2	1 : 1
0	0	1	1 : 4	1 : 2
0	1	0	1 : 8	1 : 4
0	1	1	1 : 16	1 : 8
1	0	0	1 : 32	1 : 16
1	0	1	1 : 64	1 : 32
1	1	0	1 : 128	1 : 64
1	1	1	1 : 256	1 : 128

Figuur 7/6.5.2-12: Het option-register.



Figuur 7/6.5.2-13: Directe/indirecte adressering.

6.5 PIC16/17 typen 8 bit microcontrollers

Option Register
Het optie-register is een 6 bit breed, write-only register dat allerlei besturingsbits bevat om de TMR0/WDT prescaler, de externe INT interrupt en TMR0 te configureren (figuur 7/6.5.2-12). Door de OPTION instructie uit te voeren wordt de inhoud van het W-register overgebracht naar het OPTION register. Door een RESET worden alle bits in het option-register op "1" gezet.

Indirecte data adressering, INDF en FSR Registers
Het INDF register is geen echt register, maar wordt samen met het FSR register gebruikt om indirect te adresseren. Indirecte adressering is mogelijk door het INDF register te gebruiken. Elke instructie die gebruik maakt van het INDF register krijgt in werkelijkheid toegang tot data die wordt aangewezen door het file-select register (FSR). Door INDF zelf uit te lezen (als bijvoorbeeld FSR = 0) wordt indirect 00h geproduceerd. Door naar het INDF register te schrijven ontstaat indirect een no-operation, hoewel er status-bits kunnen zijn beïnvloed.
In VOORBEELD 1 is een eenvoudig programma gegeven om RAM-lokatie 20h-2Fh te clearen, waarbij gebruik wordt gemaakt van indirecte adressering.

File Select Register (FSR)
Het FSR is óf 5 bit breed (PIC16C54/CR54/C55/C56) óf 7 bit (PIC16CR57). Het wordt samen met het INDF-register gebruikt om het data-geheugengebied indirect te adresseren.

De FSR<4:0> bits vormen de pointer voor de data-geheugenadressen 00h tot 1Fh. FSR<4:0> toggelt tussen de 16 laagste (00h-0Fh) en 16 hoogste (10h-1Fh) registerfiles (zie ook figuur 7/6.5.2-13). Als FSR<4> gecleared is, wijst hij naar de laagste 16 registerfiles en als hij gezet is naar de hoogste. FSR<3:0> vormt het adres van het specifieke register binnen elke groep van 16.
Wanneer niet indirect wordt geadresseerd kan het FSR worden gebruikt als een 5 bit (FSR<4:0>) breed algemeen register. Dit wordt echter niet aangeraden.

- PIC16C54/CR54/C55/C56
  Gebruikt geen banking. FSR<7:5> zijn niet geïmplementeerd en worden als "1" gelezen.
- PIC16C57
  FSR<6:5> zijn de bank-select bits (00 = bank 0, 01 = bank 1, 10 = bank 2, 11 = bank 3). De laagste 16 registerfiles voor de banken 1, 2 en 3 worden naar bank 0 gemapt en zijn niet toegankelijk. Met andere woorden: de FSR<6:5> bits worden genegeerd als FSR<4> gecleared is (=0). FSR<7> is niet geïmplementeerd en wordt altijd als "1" gelezen.

Program Counter
De programma-teller genereert de adressen voor maximaal 2.048 x 12 on-chip EPROM/ROM cellen die de programma-instructiewoorden bevatten (figuur 7/6.5.2-9). Afhankelijk van het PIC-type zijn de program-counter en de bijbehorende hardware-stack 9 of 11 bit breed.

VOORBEELD 1

```
        movlw   0x10      ; initialize pointer
        movwf   FSR       ; to RAM
Next    clrf    INDF      ; clear lcc
        incf    FSR       ; increment pntr
        btfsc   FSR,4     ; all done?
        goto    Next      ; no, clear next location
```

6.5 PIC16/17 typen 8 bit microcontrollers

Part #	PC width	Stack width
PIC16C54\CR54\C55	9-bit	9-bit
PIC16C56	10-bit	10-bit
PIC16C57	11-bit	11-bit

Tabel 7/6.5.2-2: Program-counter en stack breedten.

De program-counter wordt met RESET op allemaal "1"-en gezet. Bij de uitvoering van het programma wordt hij bij elke instructie automatisch met één verhoogd, tenzij het resultaat van die instructie de PC zelf wijzigt.

- De GOTO instructie maakt direct laden van de laagste 9 program-counter bits (PC<8:0>) mogelijk.
  In het geval dat het programma-geheugen groter is dan 512 bytes worden de hoogste twee bits (PC<10:9>) geladen met de pagina-select bits PA1:PA0 (STATUS<6:5>). GOTO maakt dus jumps naar elke lokatie op elke pagina mogelijk.
- De CALL instructie laadt de laagste 8 bits van de PC direct, terwijl het 9e bit naar "0" wordt gecleared. De PC-waarde (met één verhoogd) wordt op de stack gezet. In gevallen dat het programma-geheugen groter is dan 512 bytes worden de hoogste 2 PC-bits (PC<10:9>) geladen met de pagina-select bits PA1:PA0 (STATUS<6:5>).
- De RETLW instructie laadt de program-counter met de inhoud van de Top Of Stack (TOS).
- Als de PC de bestemming is van een instructie (bijvoorbeeld MOVWF PC, ADDWF PC of BSF PC, 5) wordt het berekende 8 bit resultaat in de laagste 8 bits van de program-counter geladen. Het 9e bit van de PC wordt gecleared. In het geval dat het programma-geheugen groter is dan 512 bytes worden de pagina-select bits PA1:PA0 (STATUS<6:5>) in de hoogste 2 PC-bits (PC<10:9>) geladen.

Merk op dat, aangezien het 9e bit van de PC (bit 8) wordt gecleared door de CALL instructie of een andere instructie die naar de PC schrijft (bijvoorbeeld MOVWF PC), alle subroutine-calls of berekende jumps beperkt zijn tot de eerste 256 lokaties van elke programma-geheugenpagina (512 woorden lang).
Het is ook mogelijk de program-counter te incrementeren wanneer deze naar het laatste adres van een geselecteerde pagina wijst. Hierdoor gaat de program-counter verder op de aanliggend hogere pagina. De pagina preselect-bits in het statusregister zullen hierdoor echter niet veranderen en de volgende GOTO, CALL, ADDWF PC of MOVWF PC instructie zal op de vorige pagina terugkeren, tenzij de pagina preselect-bits onder programmabesturing in orde werden gemaakt. Een NOP op lokatie 1FFh (pagina 0) bijvoorbeeld incrementeert de PC tot 200h (pagina 1). Een GOTO xxx op 200h brengt het programma terug naar adres xxxh op pagina 0 (aangenomen dat PA1:PA0 gecleared zijn). Na een RESET conditie wordt pagina 0 gepreselecteerd, terwijl de program-counter de laatste lokatie in de laatste pagina adresseert.
Zodoende zal een GOTO instructie op deze lokatie de program-counter automatisch laten verder gaan in pagina 0.

W(erk) Register
Het W-register bevat de tweede operand in twee-operand instructies en/of ondersteunt de interne data-overdracht.

Time-Out en Power Down statusbits ($\overline{TO}$, $\overline{PD}$)
De $\overline{TO}$ en $\overline{PD}$ bits in het statusregister kunnen worden getest om te bepalen of een RESET conditie werd veroorzaakt door een Watchdog Timer time-out, een power-up conditie of een wake-up from SLEEP door de Watchdog Timer of $\overline{MCLR}$-pin. Deze statusbits worden alleen beïnvloed door gebeurtenissen die in tabel 7/6.5.2-3 zijn vermeld.

6.5 PIC16/17 typen 8 bit microcontrollers

Event	$\overline{TO}$	$\overline{PD}$	Remarks
Power-up	1	1	
WDT Timeout	0	x	No effect on $\overline{PD}$
SLEEP instruction	1	0	
CLRWDT instruction	1	1	

A WDT timeout will occur regardless of the status of the $\overline{TO}$ bit. A SLEEP instruction will be executed, regardless of the status of the $\overline{PD}$ bit.

Tabel 7/6.5.2-3: Gebeurtenissen die de $\overline{TO}/\overline{PD}$ statusbits beïnvloeden.

$\overline{TO}$	$\overline{PD}$	RESET was caused by
0	0	WDT wake-up from SLEEP
0	1	WDT time-out (not during SLEEP)
1	0	$\overline{MCLR}$ wake-up from SLEEP
1	1	Power-up
u	u	= Low pulse on $\overline{MCLR}$ input

The $\overline{TO}$ and $\overline{PD}$ bits maintain their status (u) until an event of Table 4-3 occurs. A low-pulse on the $\overline{MCLR}$ input does not change the $\overline{TO}$ and $\overline{PD}$ status bits.

Tabel 7/6.5.2-4: $\overline{TO}/\overline{PD}$ status na resetten.

I/O-poorten

Inleiding

Net als bij alle andere registers kan in de I/O-registers worden gelezen en geschreven onder programma-besturing. Leesinstructies (zoals MOVF PORTB, W) lezen echter altijd de I/O-pennen, ongeacht de input/output modes ervan. Na een RESET zijn alle I/O-poorten gedefinieerd als ingang (uitgangen zijn hoog-impedant) aangezien de I/O besturingsregisters (TRISA, TRISB, TRISC) allemaal gezet zijn.

PORTA

PORTA is een 4 bit I/O-register. Alleen de laagste 4 bits worden gebruikt (RA3:RA0). De bits 7-4 zijn niet geïmplementeerd en worden als "0" uitgelezen.

PORTB

PORTB is een 8 bit I/O-register (PORTB<7:0>).

PORTC

- PIC16C55/C57:
  8 bit I/O-register
- PIC16C54/CR54/C56:
  General Purpose Register

TRIS Registers

De output driver besturingsregisters worden, door uitvoering van de TRIS f instructie, geladen met de inhoud van het W-register. Een "1" van een TRIS-registerbit zet de overeenkomende uitgangsdriver in een hoog-impedante toestand. Een "0" zet de inhoud van de uitgangs-datalatch op de geselecteerde pennen.

Let op: een lees-operatie van de poorten leest de pennen, NIET de uitgangs-datalatches. Dit wil zeggen dat, als een uitgangsdriver op een pen is vrijgegeven (enabled) en HOOG wordt gestuurd, maar door het externe systeem LAAG wordt gehouden, een lees-operatie van deze poort zal aangeven dat de pen LAAG is.

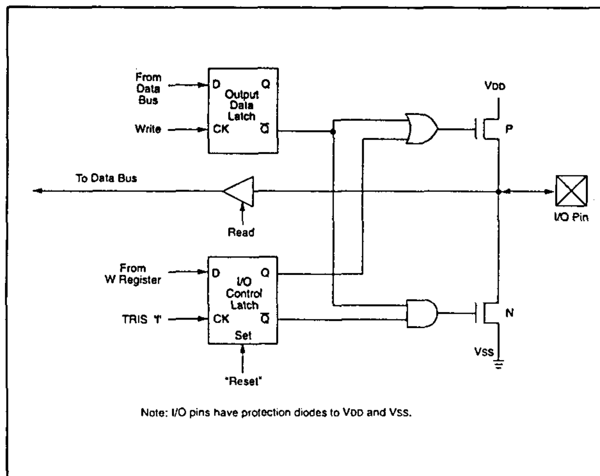
De TRIS-registers zijn "write-only" en worden na een RESET met allemaal "1"-en gevuld (output drivers disabled).

I/O Interfacing

De equivalente schakeling van een I/O-poort is te zien in figuur 7/6.5.2-14. Alle poorten kunnen worden gebruikt voor zowel ingangs- als uitgangs-operaties.

Voor ingangs-operaties zijn deze poorten niet-latchend. Een ingangssignaal moet aanwezig blijven totdat het is ingelezen door een input-instructie (bijvoorbeeld MOVF PORTB, W). De uitgangen zijn gelatched en blijven onveranderd totdat de uitgangslatch is overschreven. Om een poort-pen als uitgang te gebruiken moet het overeenkomstige direction-controlbit (in TRISA, TRISB of TRISC) worden gecleared (= "0"). Voor gebruik als ingang moet het overeenkomstige TRIS-bit "1" zijn. Elke I/O-pen kan individueel als in-

gang of als uitgang worden geprogrammeerd.



Figuur 7/6.5.2-14: Equivalente schakeling van een enkele I/O-pen.

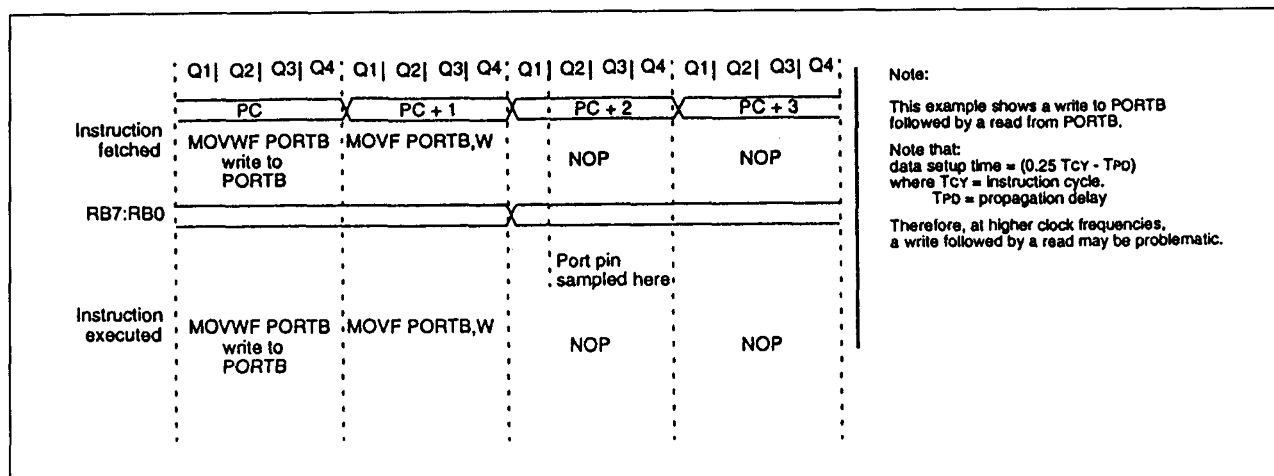
I/O programmeer-overwegingen bij bi-directionele I/O-poorten

Sommige instructies werken intern als lees-, gevolgd door schrijf-operaties. De BCF en BSF instructies lezen de gehele poort bijvoorbeeld in de CPU, voeren de bit-operatie uit en zetten het resultaat weer op de uitgang. Er moet voorzichtig gehandeld worden als deze instructies worden toegepast op een poort waarvan één of meerdere pennen als in- of uitgang worden gebruikt. Een BSF operatie op bit 5 van PORTB maakt bijvoorbeeld dat alle acht bits van PORTB in de CPU worden ingelezen. Daarna vindt de BSF-operatie op bit 5 plaats en de PORTB-waarde wordt op de uitgangslatches gezet. Als nu een ander bit van PORTB (bijvoorbeeld bit 0) als bi-directionele I/O-pen wordt gebruikt en op dat moment als ingang is gedefinieerd, zal het ingangssignaal op de pen in de CPU worden gelezen en naar de datalatch van deze pen worden geschreven, waarbij de vorige inhoud wordt overschreven. Zolang de pen in de ingangs-mode blijft, is er niets aan de hand. Als bit 0 later echter in de uitgangs-mode wordt gezet, is de inhoud van de datalatch onbekend. VOORBEELD 2 laat het effect van twee opvolgende read-modify-write instructies (bijvoorbeeld BCF, BSF, enz.) op een I/O-poort zien. Een pen met een aktief LAGE of HOGE uitgang moet op dat moment niet door externe schakelingen worden aangedreven omdat daardoor de chip beschadigd kan worden.

VOORBEELD 2

```
; Initial PORT Settings
;                  PORTB<7:4> Inputs
;                  PORTB<3:0> Outputs
; PORTB<7:6> hebben externe pull-ups en zijn
; niet met andere schakelingen verbonden
;
;                                  PORT latch     PORT pins
        BCF   PORTB, 7           ; 01pp pppp      11pp pppp
        BCF   PORTB, 6           ; 10pp pppp      11pp pppp
        BSF   STATUS,RPO         ;
        MOVLW 03Fh               ;
        TRIS  PORTB              ; 10pp pppp      10pp pppp
;
; Merk op dat de gebruiker verwacht kan hebben dat de
; pen-waarden 00pp pppp zouden zijn. Door de tweede BCF
; werd RB7 als de pen-waarde gelatched (HOOG).
```

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-15: Opeenvolgende I/O-operaties.

Opeenvolgende operaties op I/O-poorten

Het feitelijke schrijven naar een I/O-poort gebeurt aan het einde van een instructiecyclus. Daarentegen moet de data aan het begin van de instructiecyclus aanwezig zijn om te kunnen lezen (zie figuur 7/6.5.2-15). Daarom moet men voorzichtig zijn als een schrijfoperatie wordt gevolgd door lezen op dezelfde I/O-poort. De instructie-volgorde dient zo te zijn dat de spanning op de pen stabiel is vóór de volgende lees-instructie. In geval van twijfel is het beter deze instructies te scheiden met een NOP of andere instructie die niet op deze I/O-poort slaat.

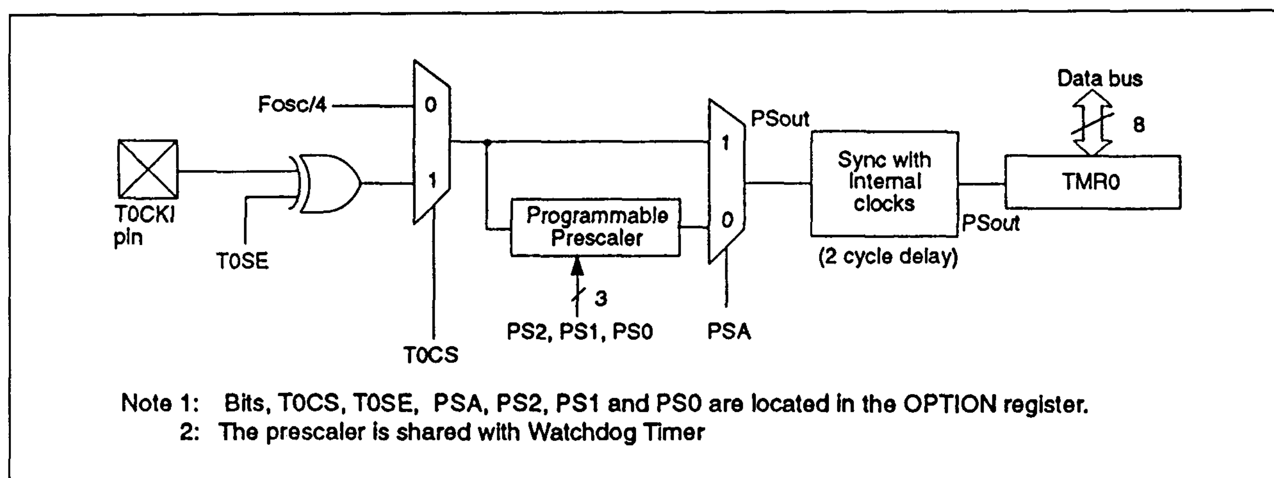
Timer0 (TMR0) Module

Inleiding

De TMR0-module timer/counter heeft de volgende eigenschappen:

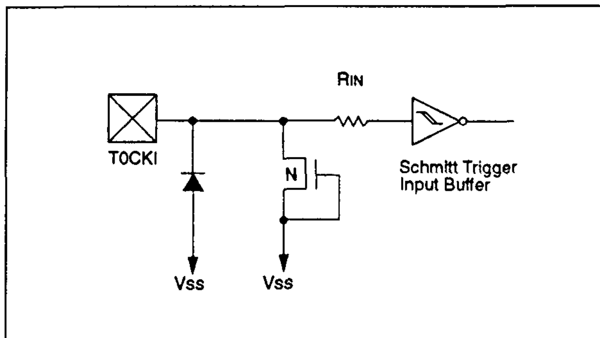
- 8 bit timer/counter
- lees- en schrijfbaar
- 8 bit met software programmeerbare prescaler
- interne of externe clock-select
- edge select voor externe clock

Figuur 7/6.5.2-16 geeft het vereenvoudigde blokschema van de TMR0-module, terwijl figuur 7/6.5.2-17 de elektrische structuur van de TMR0 ingang laat zien.



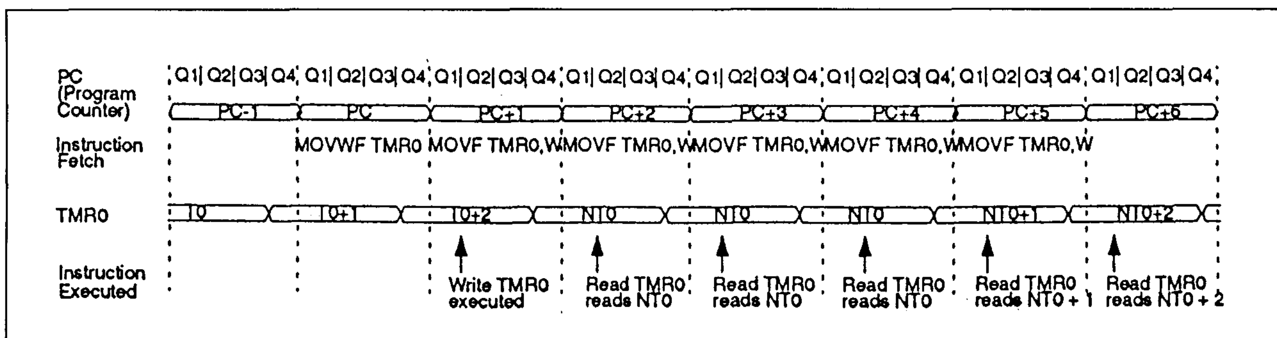
Figuur 7/6.5.2-16: Vereenvoudigd blokschema van TMR0.

6.5 PIC16/17 typen 8 bit microcontrollers

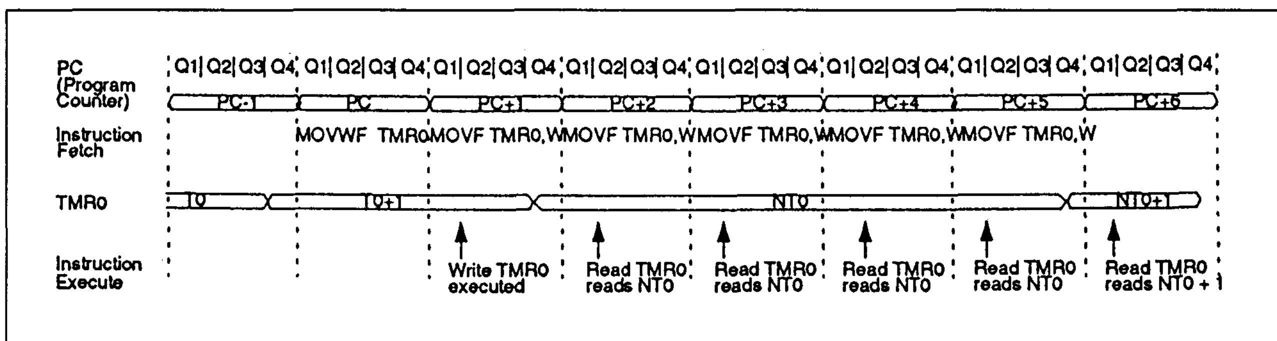


Figuur 7/6.5.2-17: Elektrische structuur van de TOCKI-pen.

De counter-mode wordt geselecteerd door het zetten van het TOCS-bit (OPTION<5>). In deze mode incrementeert TMR0 op elke stijgende of dalende flank van het signaal op TOCKI. De flank wordt bepaald door het T0 source edge (TOSE) select-bit (OPTION<4>). Door het TOSE-bit te clearen wordt de stijgende flank geselecteerd.
De prescaler wordt gedeeld door de TMR0-module en de watchdog timer. De instelling van de prescaler wordt geregeld door het besturingsbit PSA (OPTION<3>).



Figuur 7/6.5.2-18: Timing van TMR0: interne clock, geen prescale.

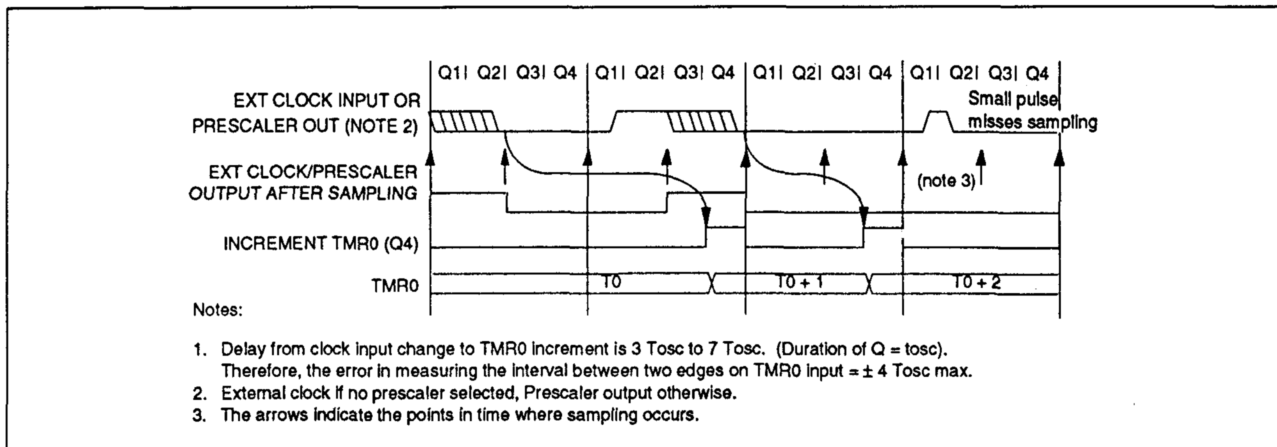


Figuur 7/6.5.2-19: Timing van TMR0: interne clock, wél prescale 1:2.

De timer-mode wordt geselecteerd door het TOCS-bit te clearen (OPTION<5>). De TMR0-module zal dan iedere instructiecyclus incrementeren (zonder prescaler). Als TMR0 wordt geschreven, wordt het incrementeren gedurende de volgende twee cycli gesperd (figuren 7/6.5.2-18 en -19). De gebruiker kan dit omzeilen door een aangepaste waarde naar de TMR0-module te schrijven.

Het zetten van het PSA-bit wijst de prescaler toe aan de WDT en maakt dat TMR0 een prescale van 1:1 krijgt.
Door clearen van het PSA-bit beschikt TMR0 over de prescaler.
De prescaler kan niet worden uitgelezen en is ook niet beschrijfbaar. Wordt de prescaler toegewezen aan de TMR0-module dan kan een prescale-waarde van 1:2, 1:4, ..., 1:256 worden ingesteld.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-20: TIMER0 timing met een externe clock.

Gebruik van TMR0 met een externe clock

Wanneer voor TMR0 een externe clock wordt gebruikt, worden daaraan bepaalde eisen gesteld om synchronisatie met de interne phase-clock (T_{OSC}) mogelijk te maken. Ook ondervindt het incrementeren van TMR0 een vertraging na de synchronisatie.

Externe clock synchronisatie

Wanneer er geen prescaler wordt gebruikt is het externe clock ingangssignaal hetzelfde als het prescaler uitgangssignaal. Voor het synchroniseren van T0CKI op de interne phase-clock is bemonstering van de prescaler-uitgang op de Q2 en Q4 cycli van de interne phase-clock's nodig (zie figuur 7/6.5.2-20). Daarom moet T0CKI tenminste gedurende 2 T_{OSC} (plus een kleine RC-vertraging) HOOG zijn en even lang LAAG. Als een prescaler wordt gebruikt, wordt het externe clock-signaal gedeeld door de prescaler (een asynchrone ripple-counter) om er zeker van te zijn dat het prescaler-uitgangssignaal symmetrisch is. Om aan de eisen voor het bemonsteren te voldoen, moet bij de externe clock rekening worden gehouden met de ripple-counter. Het is daarom nodig dat T0CKI een periode heeft van tenminste 4 T_{OSC} (+ RC-vertraging), gedeeld door de prescaler-waarde. De enige beperking op de HOOG- en LAAG-tijd van T0CKI is dat ze groter zijn dan de minimale pulsbreedte van 10 ns.

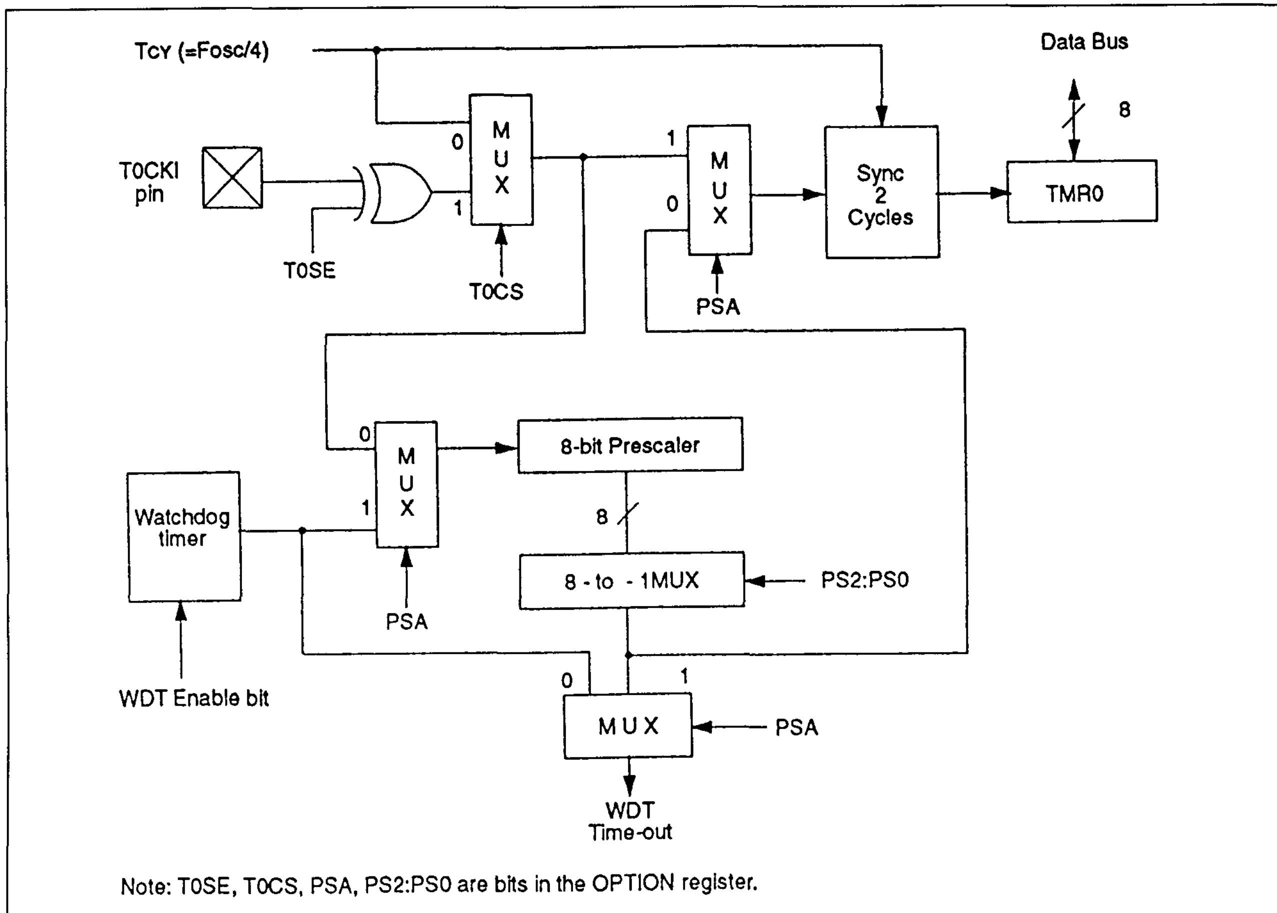
TMR0 increment vertraging

Aangezien het uitgangssignaal van de prescaler op de interne clocks wordt gesynchroniseerd, is er een kleine vertraging tussen de tijd dat de externe clock-flank optreedt en de tijd dat de TMR0-module feitelijk incrementeert. In figuur 7/6.5.2-20 is deze vertraging te zien.

Prescaler

Er is een 8 bit teller beschikbaar als prescaler voor de TMR0-module of als postscaler voor de Watchdog Timer. Voor de eenvoud wordt deze teller hier overal "prescaler" genoemd. Omdat er slechts één prescaler is, betekent dit dat als hij is toegewezen aan de TMR0-module er geen prescaler is voor de Watchdog Timer, en omgekeerd. De PSA en PS2:PS0 bits bepalen de toewijzing van de prescaler en de prescaleverhouding (OPTION<3:0>). Bij toewijzing aan de TMR0-module zullen alle instructies die naar de TMR0-module schrijven (bijvoorbeeld CLRF 1, MOVWF 1, BSF 1,x...enz.) de prescaler clearen. Bij toewijzing aan de WDT cleart een CLRWDT instructie de prescaler tegelijk met de Watchdog Timer. De prescaler kan niet worden uitgelezen of beschreven. Na een RESET bevat de prescaler allemaal "0"-en.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-21: Blokschema van de TMR0/WDT prescaler.

VOORBEELD 3: prescaler-wisseling (TMR0WDT)

```
CLRF    TMR0              ; clear TMR0
CLRWDT                    ; clear WDT en prescaler
MOVLW 'xxxx1xx'b          ; selecteer nieuwe prescale-waarde
OPTION                    ;
```

Omschakelen van de prescaler-toewijzing

De toewijzing van de prescaler geschiedt geheel onder software-besturing en kan dus "on the fly" tijdens de uitvoering van een programma worden gewijzigd. Om een onbedoelde RESET te voorkomen moet de instructie-volgorde van VOORBEELD 3 worden uitgevoerd bij het wisselen van de prescaler tussen TMR0 en WDT.

Om de prescaler van WDT over te laten gaan naar TMR0 moet de volgorde van VOORBEELD 4 worden toegepast (ook als de WDT is uitgeschakeld). Merk op dat een CLRWDT instructie moet worden uitgevoerd vóór omschakelen van de prescaler.

6.5 PIC16/17 typen 8 bit microcontrollers

VOORBEELD 4: prescaler-wisseling (WDTTMR0)

```
CLRWDT              ; clear WDT en prescaler
MOVLW 'xxxx0xx'b    ; selecteer TMR0, nieuwe
                      prescale-waarde en clock-source
OPTION              ;
```

Register Name	Function	Address	Power-on Reset Value
TMR0	Timer/counter register	01h	xxxx xxxx
OPTION	Configuration and prescaler assignment bits for TMR0. (Figure 4-4)	N/A	--11 1111

Legend: x = unknown, - = unimplemented, read as '0'.

Tabel 7/6.5.2-5: Samenvatting van de TMR0 registers.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01	TMR0	Timer0 - 8-bit real-time clock counter							
N/A	OPTION	—	—	TOCS	TOSE	PSA	PS2	PS1	PS0

Legend: — = Unimplemented, read as '0'.
Note: Shaded cells are not used by TMR0 module.

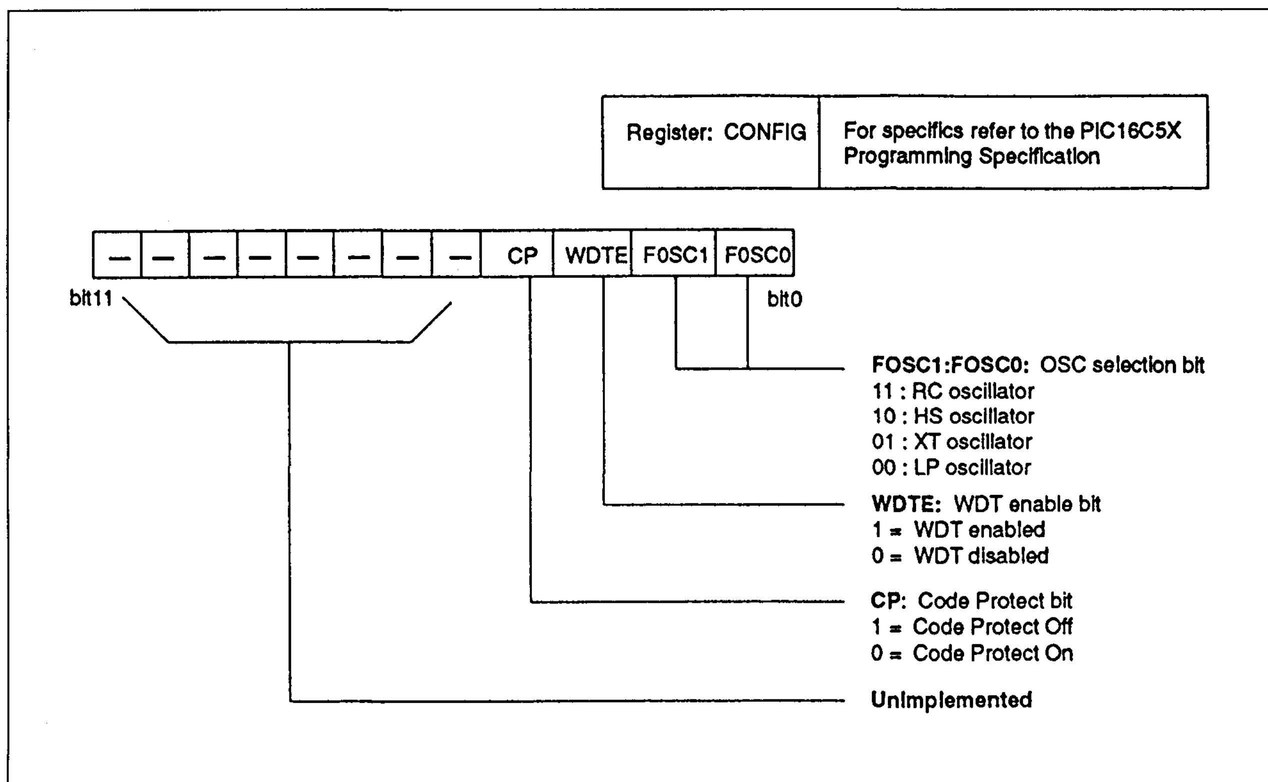
Tabel 7/6.5.2-6: Registers die met TMR0 samenwerken.

Speciale mogelijkheden van de CPU

Inleiding

De PIC16C5x-familie beschikt over een uitgebreide reeks speciale voorzieningen die tegemoet komen aan de eisen die in realtime toepassingen worden gesteld. Deze voorzieningen zijn reeds vermeld bij de algemene gegevens en worden hier verder uitgewerkt. De Watchdog Timer kan alleen worden uitgeschakeld met het WDTE configuratiebit. De Device Reset Timer (DRT) voorziet in een vertraging van 18 ms om de chip gereset te houden totdat de kristaloscillator stabiel is. Door deze on-chip timer is voor de meeste toepassingen geen externe reset-schakeling nodig.
De SLEEP mode is een power-down mode, waarbij zeer weinig energie wordt verbruikt. Om hieruit te ontwaken kan een externe reset of een watchdog timer time-out worden gebruikt. Ook staan verschillende oscillatoropties ter beschikking om de PIC16C5x geschikt te maken voor de toepassing. De RC-oscillator optie is goedkoper, terwijl de LP kristal-optie energie bespaart. Door middel van een set configuratiebits kan uit verschillende mogelijkheden worden gekozen.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-22: Het configuratiewoord van de PIC16C5x.

Configuratiebits

Het configuratiewoord bestaat uit 4 of 12 bits, afhankelijk van de configuratie van de PIC16C5x (zie figuur 7/6.5.2-22). Twee van de programmeerbare configuratiebits dienen voor het selecteren van het type oscillator, één is het Watchdog Timer enable-bit en één is het code-beveiligingsbit.

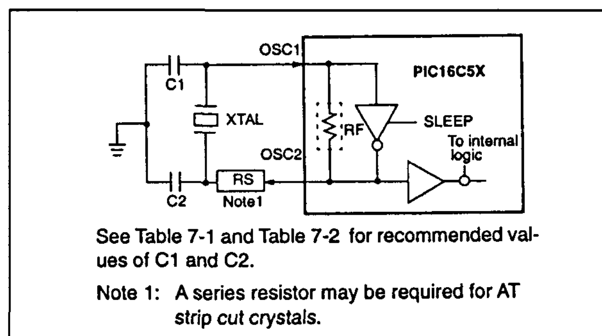
Oscillator-configuraties

De PIC16C5x kan met vier verschillende oscillator-modes werken. De gebruiker kan één van de volgende modes bij de fabriek bestellen:

- LP: Low Power Crystal;
- XT: Crystal/Resonator;
- HS: High Speed Crystal/Resonator;
- RC: Resistor/Capacitor.

Crystal Oscillator/Ceramic Resonator

In de XT, LP of HS mode is een kristal of een ceramische resonator verbonden met de OSC1/CLKIN en OSC2/CLKOUT pennen om (parallel) oscilleren te veroorzaken (zie figuur 7/6.5.2-23). In de tabellen 7/6.5.2-7 en -8 zijn de bijbehorende waarden voor de condensatoren opgenomen. Bovendien is in de XT, LP of HS mode een externe clock-aandrijving op OSC1/CLKIN mogelijk (figuur 7/6.5.2-24).



See Table 7-1 and Table 7-2 for recommended values of C1 and C2.

Note 1: A series resistor may be required for AT strip cut crystals.

Figuur 7/6.5.2-23: Oscillator met kristal of ceramische resonator (HS, XT of LP OSC configuratie).

6.5 PIC16/17 typen 8 bit microcontrollers

Osc Type	Resonator Freq	Cap.Range C1	Cap. Range C2
LP	32 kHz†	33-68 pF	33-68 pF
	200 kHz	15-33 pF	15-33 pF
XT	100 kHz	68-100 pF	68-100 pF
	2 MHz	10-22 pF	10-22 pF
	4 MHz	10-22 pF	10-22 pF
HS	8 MHz	22-47 pF	22-47 pF
	20 MHz	22-47 pF	22-47 pF

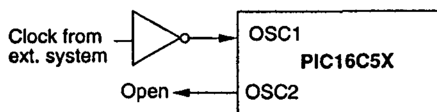
† For VDD > 4.5V, C1 = C2 ≈ 30 pF is recommended.
These values are for design guidance only. Rs may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

Tabel 7/6.5.2-7: Condensator-waarden voor een kristal-oscillator.

Osc Type	Resonator Freq	Cap. Range C1	Cap. Range C2
XT	455 kHz	68-100 pF	68-100 pF
	2.0 MHz	15-68 pF	15-68 pF
	4.0 MHz	10-100 pF	10-100 pF
HS	8.0 MHz	10-68 pF	10-68 pF
	16.0 MHz	10-10 pF	10-10 pF

These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for appropriate values of external components.

Tabel 7/6.5.2-8: Condensator-waarden voor oscilleren met een ceramische resonator.

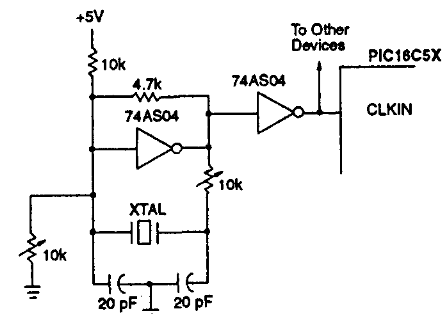


Figuur 7/6.5.2-24: Oscillator met externe clock-aansturing (HS, XT of LP OSC configuratie).

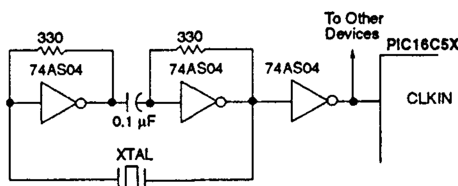
Externe oscillator-schakeling

In plaats van een kristal/resonator kan ook een kant-en-klare oscillator of een eenvoudige, uit TTL-poorten opgebouwde oscillator worden gebruikt. Kant-en-klare oscillatoren werken in een uitgebreid temperatuurgebied en zijn zeer stabiel, terwijl zelf vervaardigde typen met TTL-poorten ook goed voldoen. Er zijn dan twee oscillator-schakelingen mogelijk: met parallel resonantie en met serie resonantie.

In figuur 7/6.5.2-25 is een parallel oscillerende schakeling te zien, die op de grondfrequentie van het kristal werkt. De 74AS04 inverter zorgt voor de benodigde 180° fasedraaiing. De 4,7 kΩ weerstand levert de negatieve terugkoppeling die voor stabiliteit nodig is. Met de 10 kΩ potentiometer wordt de 74AS04 in het lineaire werkgebied gezet.

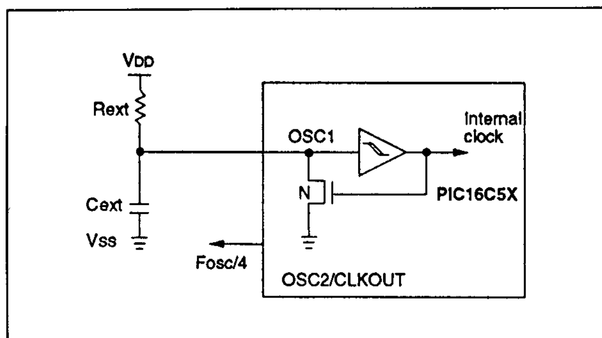


Figuur 7/6.5.2-25: Externe oscillator met parallelle oscillatie.



Figuur 7/6.5.2-26: Externe oscillator met seriële oscillatie.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-27: RC-oscillator mode.

$\overline{TO}$	$\overline{PD}$	RESET was caused by
0	0	WDT wake-up from SLEEP
0	1	WDT time-out (not during SLEEP)
1	0	$\overline{MCLR}$ wake-up from SLEEP
1	1	Power-up
u	u	= Low pulse on $\overline{MCLR}$ input

The $\overline{TO}$ and $\overline{PD}$ bit maintain their status (u) until a reset occurs. A low-pulse on the $\overline{MCLR}$ input does not change the $\overline{TO}$ and $\overline{PD}$ status bits.

Tabel 7/6.5.2-9: Status van $\overline{TO}/\overline{PD}$ na RESET.

Figuur 7/6.5.2-26 toont een in serie oscillerende schakeling, die ook op de grondfrequentie van het kristal werkt. De inverter verzorgt een 180° fasedraaiing in een serieresonante schakeling. De 330 Ω weerstand levert de negatieve terugkoppeling om de inverters in hun lineaire werkgebied te zetten.

RC-oscillator

Voor toepassingen waarbij de timing niet zo belangrijk is, is de RC-optie een goede keuze. De RC oscillator-frequentie is een functie van de voedingsspanning, de waarden van weerstand (R_{ext}) en condensator (C_{ext}) en de bedrijfstemperatuur. Afgezien daarvan zal de oscillator-frequentie van eenheid tot eenheid afwijken als gevolg van de normale proces-parameter variaties. Bovendien zal de (meetellende) capaciteit van de aansluitdraden per behuizingstype verschillend zijn, vooral bij kleine C_{ext}-waarden. De gebruiker moet ook rekening houden met afwijkingen door de toleranties van de gebruikte R en C componenten. In figuur 7/6.5.2-27 is een RC-oscillator in samenwerking met een PIC16C5x te zien. Voor kleinere waarden van R_{ext} dan 2,2 kΩ kan de oscillator instabiel worden of zelfs helemaal stoppen. Voor zeer grote R_{ext}-waarden (bijvoorbeeld 1 MΩ) wordt de oscillator gevoelig voor storingen, vocht en lek. Er wordt dus aanbevolen R_{ext} tussen 3 kΩ en 100 kΩ te houden.
Hoewel de oscillator ook zonder externe condensator werkt, worden waarden boven 20 pF aanbevolen voor betere stabiliteit en storingsongevoeligheid. Zonder of met een zeer kleine externe condensator kan de oscillatie-frequentie drastisch variëren als gevolg van veranderingen in de externe capaciteit (bijvoorbeeld van printsporen of aansluitdraden). De oscillatie-frequentie, gedeeld door 4, is beschikbaar op de OSC2/CLKOUT-pen en kan worden gebruikt voor test-doeleinden of voor het synchroniseren van andere logika.

Reset

De PIC16C5x maakt onderscheid tussen verschillende soorten reset:
- Power-On Reset (POR);
- $\overline{MCLR}$ reset tijdens normaal bedrijf;
- $\overline{MCLR}$ reset tijdens SLEEP;
- WDT time-out reset.

Sommige registers worden door geen enkele reset-conditie gereset. De status daarvan is onbekend bij POR en onveranderd bij andere reset's. De meeste andere registers worden door POR, $\overline{MCLR}$ of een WDT reset in een "reset toestand" gebracht. Merk op dat de PIC16C5x geen onderscheid maakt tussen een WDT reset tijdens SLEEP of tijdens normaal bedrijf. De $\overline{TO}$ en $\overline{PD}$ bits worden, afhankelijk van de verschillende reset's, gezet of gecleared (zie tabel 7/6.5.2-9). Met deze bits kan de soort reset worden bepaald (zie de tabellen 7/6.5.2-10 en -11 voor een compleet overzicht van de reset-toestanden van alle registers, respectievelijk de speciale registers). In figuur 7/6.5.2-28 is een vereenvoudigd blokschema van de on-chip resetschakeling te zien.

6.5 PIC16/17 typen 8 bit microcontrollers

Register	Address	Power-On Reset	$\overline{MCLR}$ or WDT Reset
W	N/A	xxxx xxxx	uuuu uuuu
TRIS	N/A	1111 1111	1111 1111
OPTION	N/A	--11 1111	--11 1111
INDF	00h	—	—
TMR0	01h	xxxx xxxx	uuuu uuuu
PCL	02h	1111 1111	1111 1111
STATUS	03h	0001 1xxx	000? ?uuu (1)
FSR	04h	xxxx xxxx	1uuu uuuu
PORTA	05h	---- xxxx	---- uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu
PORTC	07h	xxxx xxxx	uuuu uuuu
General Purpose register files	08-7Fh	xxxx xxxx	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented read as '0', ? = value depends on condition.

Tabel 7/6.5.2-10: Overzicht van de reset-condities van alle registers.

Condition	STATUS Addr: 03h	PCL Addr: 02h
Power-On Reset	0001 1xxx	1111 1111
$\overline{MCLR}$ reset during normal operation	000u uuuu(1)	1111 1111
$\overline{MCLR}$ reset during SLEEP	0001 0uuu	1111 1111
WDT reset during SLEEP	0000 0uuu	1111 1111
WDT reset during normal operation	0000 1uuu	1111 1111

Legend: u = unchanged, x = unknown, - = unimplemented read as '0'.
Note 1: $\overline{TO}$ and $\overline{PD}$ bits retain their last value until one of the other reset conditions occur.
2: The CLRWDT instruction will set the $\overline{TO}$ and $\overline{PD}$ bits.

Tabel 7/6.5.2-11: Reset-condities van de speciale registers.

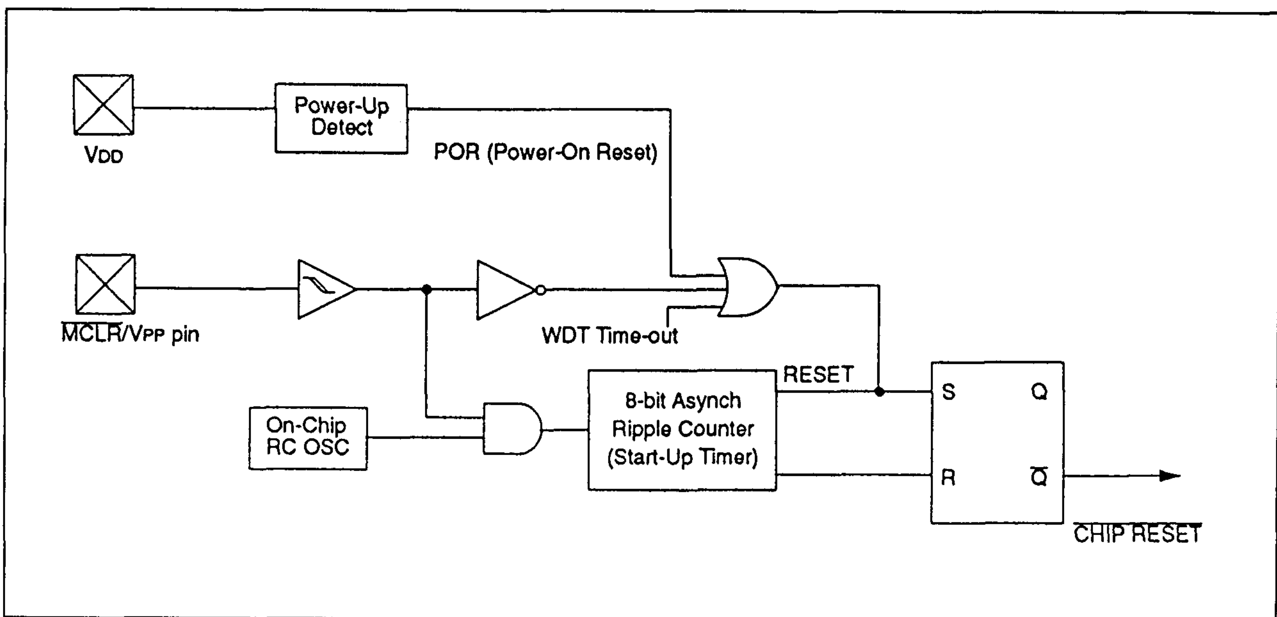
Power-On Reset (POR)

De PIC16C5x-familie heeft een Power-On Reset schakeling aan boord, waarmee voor de meeste power-up situaties een interne chip-reset wordt geleverd. Om gebruik te maken van deze eigenschap behoeft de gebruiker slechts de $\overline{MCLR}/V_{PP}$-pen aan V_{DD} te leggen (in figuur 7/6.5.2-33 is de elektrische structuur van de TMR0-ingangen te zien). Figuur 7/6.5.2-28 toont het vereenvoudigde blokschema van de on-chip reset-schakeling. De Power-On Reset-schakeling en de Device Reset Timer zijn nauw aan elkaar verwant. Bij power-up wordt de reset-latch gezet en de DTR gereset. De DTR-timer begint met tellen zodra hij detecteert dat $\overline{MCLR}$ HOOG is. Na de time-out periode (typ. 18 ms) reset hij de reset-latch en het on-chip reset-signaal.

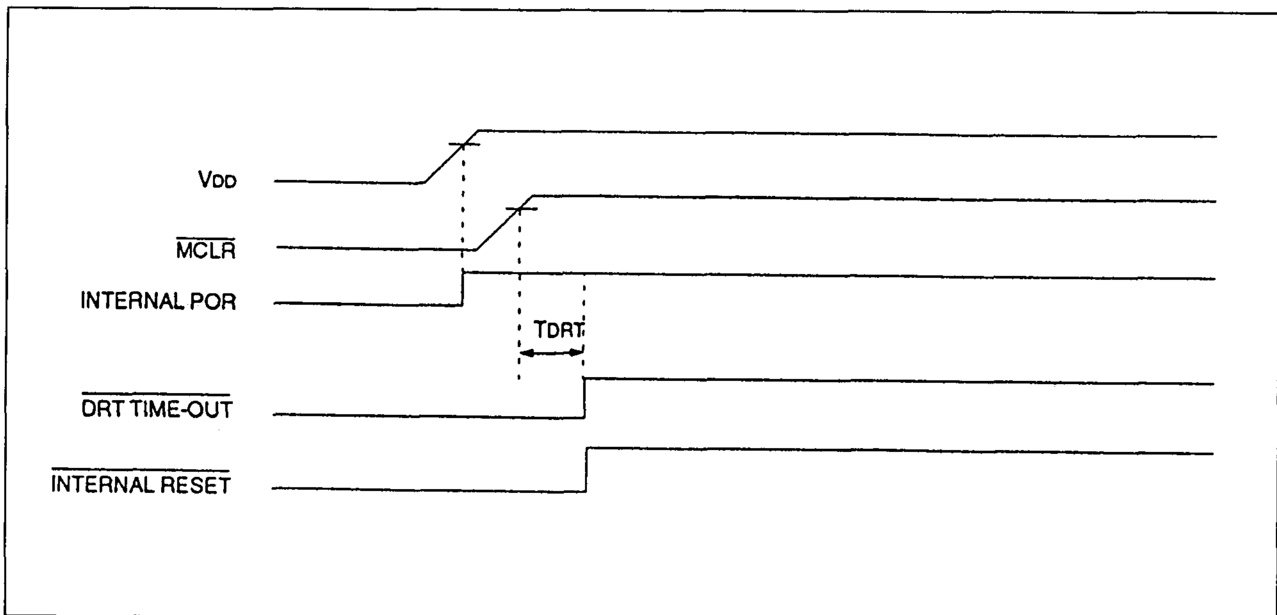
In de figuren 7/6.5.2-29 en -30 zijn twee power-up situaties te zien bij een relatief snelle opkomst van V_{DD}. In figuur 7/6.5.2-29 heeft V_{DD} de gelegenheid gehad om op te

komen en te stabiliseren voordat $\overline{MCLR}$ HOOG ging. De chip komt dan werkelijk T_{DRT} ms na het HOOG gaan van $\overline{MCLR}$ uit de reset.

In figuur 7/6.5.2-30 wordt de Power-On Reset eigenschap gebruikt ($\overline{MCLR}$ en V_{DD} zijn NIET met elkaar verbonden). Doordat V_{DD} stabiel wordt voordat de start-up timer een time-out geeft, is het resetten probleemloos.

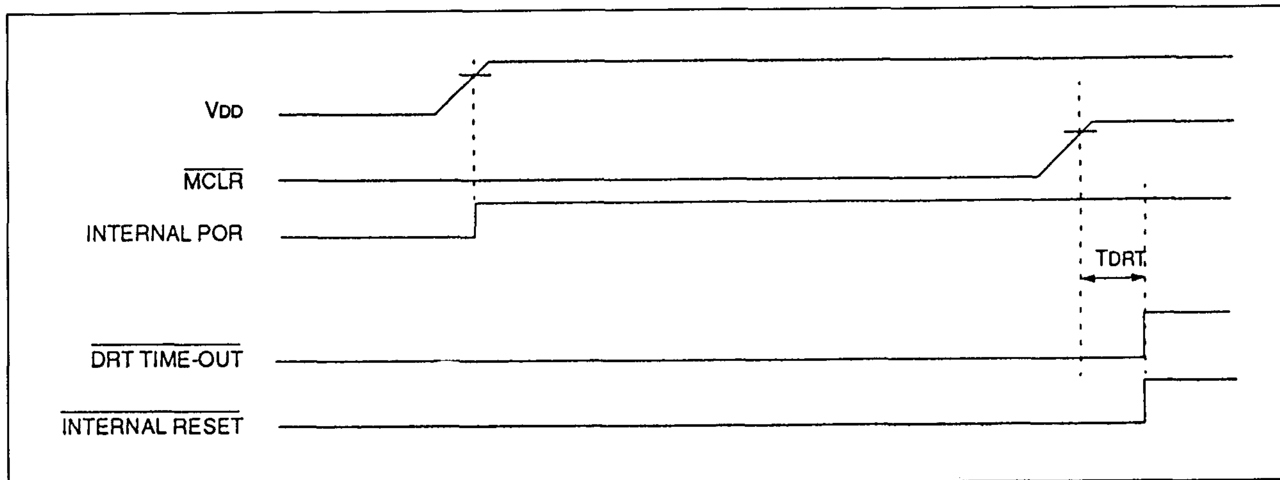


Figuur 7/6.5.2-28: Vereenvoudigd blokschema van de on-chip reset schakeling.

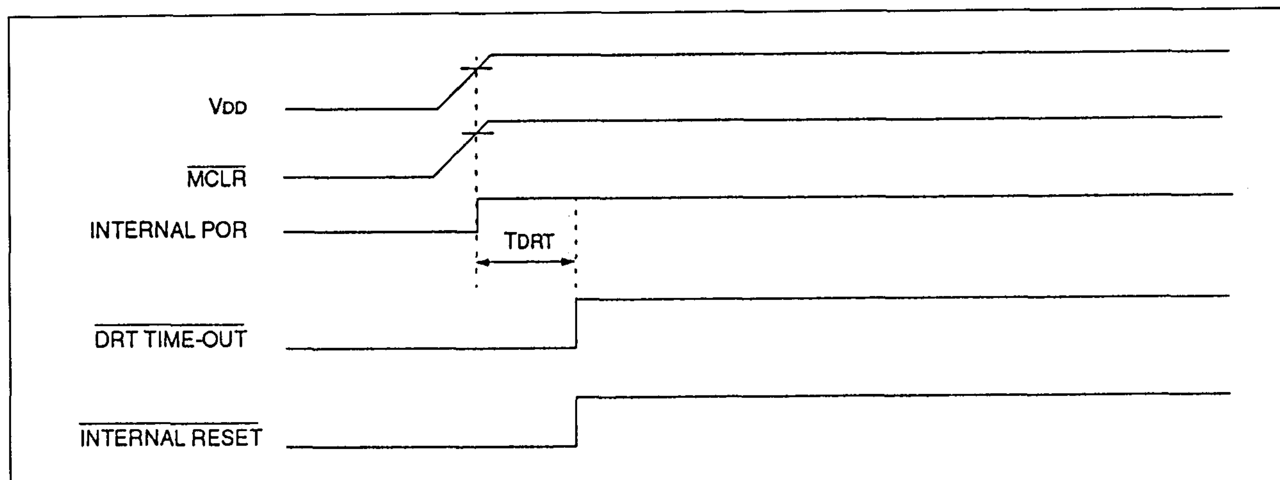


Figuur 7/6.5.2-29: Time-out volgorde bij Power-up ($\overline{MCLR}$ NIET verbonden met V_{DD}): geval 1.

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-30: Time-out volgorde bij Power-up (MCLR NIET verbonden met VDD): geval 2.

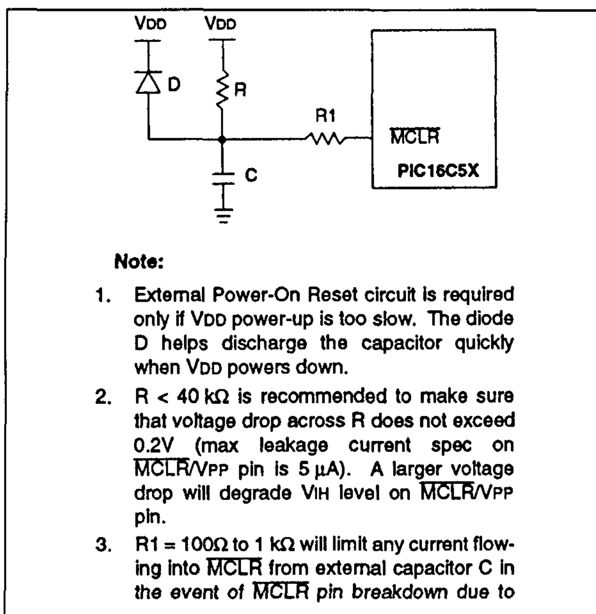


Figuur 7/6.5.2-31: Time-out volgorde bij Power-up (MCLR verbonden met VDD).

Figuur 7/6.5.2-31 voorspelt een potentieel gevaarlijke situatie, waarbij V_{DD} te langzaam HOOG gaat. In deze situatie, als de start-up timer een time-out geeft, heeft V_{DD} nog niet de $V_{DD(min)}$ waarde bereikt en kan niet gegarandeerd worden dat de chip correct werkt.
Samenvattend kan worden gezegd dat de on-chip POR gegarandeerd goed werkt als V_{DD} niet langzamer opkomt dan 0,05 V/ms en V_{DD} bij 0 V begint. De on-chip POR vertraging is te kort voor laagfrequente kristallen die veel meer dan 18 ms nodig hebben om te starten en te stabiliseren. In zulke gevallen wordt aanbevolen externe RC-schakelingen te gebruiken om langere POR-vertragingen te bewerkstelligen.

Device Reset Timer (DRT)
De Device Reset Timer levert een vaste nominale time-out van 18 ms na reset. De DRT werkt met een interne RC-oscillator. De processor wordt RESET gehouden zolang de DRT aktief is. De DRT vertraging maakt dat de V_{DD} boven $V_{DD(min)}$ kan komen en dat de oscillator stabiliseert.
Bij oscillator-schakelingen die gebaseerd zijn op kristallen of ceramische resonatoren moet na het inschakelen van de voedingsspanning een bepaalde tijd worden gewacht

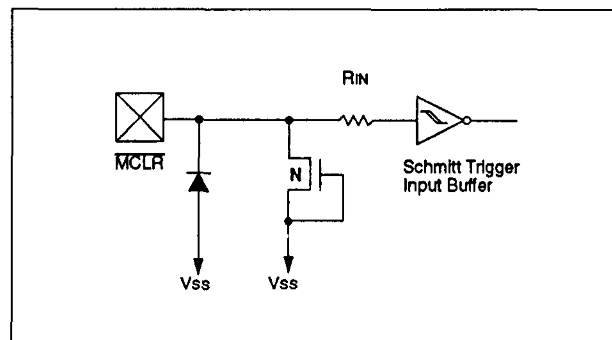
om het oscilleren stabiel te laten worden. De on-chip DRT houdt de PIC16C5x tot ongeveer 18 ms nadat de $\overline{MCLR}/V_{PP}$-pen een logisch HOOG niveau heeft bereikt in een RESET conditie. Daardoor zijn gewoonlijk geen externe RC-netwerken op de $\overline{MCLR}$-ingang nodig (figuur 7/6.2.-32). De Device Reset tijdvertraging zal van chip tot chip anders zijn, afhankelijk van V_{DD}, temperatuur en proces-variabelen. De DRT zal ook worden getriggerd bij een Watchdog Timer time-out. Dit is vooral belangrijk bij toepassingen die de WDT gebruiken om de PIC16C5x automatisch uit de SLEEP-mode te laten ontwaken.



Figuur 7/6.5.2-32: Externe reset-schakeling (voor langzame V_{DD} Power-Up).

Watchdog Timer (WDT)

De watchdog timer is gerealiseerd als een vrijlopende on-chip RC-oscillator die geen externe componenten nodig heeft. Deze RC-oscillator is een andere dan die op de OSC1/CLKIN-pen. Dit betekent dat de WDT ook zal lopen als de clock op de OSC1/CLKIN en OSC2/CLKOUT pennen is gestopt, door bijvoorbeeld de uitvoering van een SLEEP-instructie. Tijdens normaal bedrijf genereert een WDT time-out een device-RESET. Als de schakeling in de SLEEP-mode verkeert, maakt een WDT time-out dat de schakeling ontwaakt en verder gaat met de normale werking. De WDT kan permanent worden uitgeschakeld door de configuratiebit WDTE als "0" te programmeren (tabel 7/6.5.2-12).



Figuur 7/6.5.2-33: Elektrische structuur van de $\overline{MCLR}/V_{PP}$-pen.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Config. Word	—	—	—	—	CP	WDTE	FOSC1	FOSC0
OPTION	—	—	T0CS	T0SE	PSA	PS2	PS1	PS0

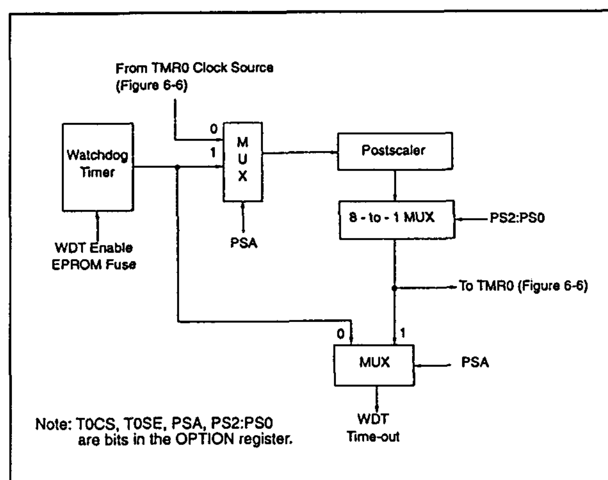
Note 1: Shaded cells are not used by the Watchdog Timer.

Tabel 7/6.5.2-12: Samenvatting van de registers die betrokken zijn met de Watchdog Timer.

6.5 PIC16/17 typen 8 bit microcontrollers

WDT-periode

De WDT heeft een nominale time-out periode van 18 ms (zonder prescaler). Deze perioden variëren met de temperatuur, V_{DD} en proces-variabelen.
Indien langere time-out perioden gewenst zijn, kan een prescaler met een deel-ratio van maximaal 1:128 worden toegekend aan de WDT (onder software besturing) door in het OPTION-register te schrijven. Dan zijn time-out perioden van maximaal 2,3 s mogelijk (zie figuur 7/6.5.2-34). De CLRWDT en SLEEP instructies clearen de WDT en de postscaler (als die aan de WDT is toegewezen) en voorkomen dat een device-RESET wordt gegeven. Het $\overline{TO}$-bit (STATUS) wordt gecleared na een WDT time-out. Onder de slechtst mogelijke omstandigheden (V_{DD} = min., temperatuur = max., max. WDT prescaler) kan het een aantal seconden duren voordat een WDT time-out optreedt.



Figuur 7/6.5.2-34: Blokschema van de Watchdog Timer.

Power-Down Mode (SLEEP)

De Power-Down mode wordt bereikt door een SLEEP-instructie uit te voeren. Als de Watchdog Timer enabled was, wordt die gecleared, maar hij blijft wel draaiend; het $\overline{TO}$-bit (STATUS<4>) wordt gezet, het $\overline{PD}$-bit (STATUS<3>) wordt gecleared en de oscillator-driver wordt uitgeschakeld. De I/O-poorten handhaven de toestand die zij hadden voordat de SLEEP-instructie werd gegeven (HOOG, LAAG of hoog-impedant). Voor het laagste stroomverbruik bij power-down moet de T0CKI-ingang op V_{DD} of V_{SS} staan, terwijl de $\overline{MCLR}/V_{PP}$-pen op logisch HOOG moet staan (V_{IHMC}).

Ontwaken uit SLEEP

De schakeling kan op één van de volgende manieren uit SLEEP ontwaken:

- een externe reset op de $\overline{MCLR}/V_{PP}$-pen;
- een watchdog timer time-out (als de WDT was vrijgegeven).

Beide gebeurtenissen veroorzaken een device-reset. De $\overline{TO}$ en $\overline{PD}$-bits kunnen worden gebruikt om te detecteren waardoor de schakeling werd gereset. Het $\overline{TO}$-bit wordt gecleared als een WDT time-out optrad (en het ontwaken veroorzaakte). Het $\overline{PD}$-bit, dat bij power-up wordt gezet, wordt bij SLEEP gecleared.
De WDT wordt bij het ontwaken uit SLEEP altijd gecleared.

Code-beveiliging

De code in het programma-geheugen kan worden beveiligd door de code-beveiligingsbits te clearen.
In een code-beveiligde mode wordt het configuratiewoord niet beveiligd, waardoor alle bits kunnen worden gelezen. Zodra de code-beveiliging is ingegaan, worden alle geheugenlokaties ge-"scrambeld" uitgelezen. Voor EPROM-schakelingen kunnen de geheugenlokaties boven 40h verder niet worden geprogrammeerd. De eerste 64 lokaties (00h tot en met 3Fh) kunnen echter wel worden geprogrammeerd. Deze lokaties worden niet "veilig" geacht.

ID-lokaties

Er zijn vier geheugenlokaties aangewezen als ID-lokaties waar de gebruiker de checksum of andere code-identificatie getallen kan opslaan. Deze lokaties zijn niet toegankelijk bij normaal bedrijf, maar zijn wel lees- en beschrijfbaar tijdens program/verify.

6.5 PIC16/17 typen 8 bit microcontrollers

Mnemonic, Operands		Description	Cycles	12-Bit Opcode			Status Affected	Notes
				MSb		LSb		
ADDWF	f,d	Add W and f	1	0001	11df	ffff	C,DC,Z	1,2,4
ANDWF	f,d	AND W with f	1	0001	01df	ffff	Z	2,4
CLRF	f	Clear f	1	0000	011f	ffff	Z	4
CLRW	-	Clear W	1	0000	0100	0000	Z	
COMF	f, d	Complement f	1	0010	01df	ffff	Z	
DECF	f, d	Decrement f	1	0000	11df	ffff	Z	2,4
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	0010	11df	ffff	None	2,4
INCF	f, d	Increment f	1	0010	10df	ffff	Z	2,4
INCFSZ	f, d	Increment f, Skip if 0	1(2)	0011	11df	ffff	None	2,4
IORWF	f, d	Inclusive OR W with f	1	0001	00df	ffff	Z	2,4
MOVF	f, d	Move f	1	0010	00df	ffff	Z	2,4
MOVWF	f	Move W to f	1	0000	001f	ffff	None	1,4
NOP	-	No Operation	1	0000	0000	0000	None	
RLF	f, d	Rotate left f through Carry	1	0011	01df	ffff	C	2,4
RRF	f, d	Rotate right f through Carry	1	0011	00df	ffff	C	2,4
SUBWF	f, d	Subtract W from f	1	0000	10df	ffff	C,DC,Z	1,2,4
SWAPF	f, d	Swap f	1	0011	10df	ffff	None	2,4
XORWF	f, d	Exclusive OR W with f	1	0001	10df	ffff	Z	2,4
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF	f, b	Bit Clear f	1	0100	bbbf	ffff	None	2,4
BSF	f, b	Bit Set f	1	0101	bbbf	ffff	None	2,4
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	0110	bbbf	ffff	None	
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	0111	bbbf	ffff	None	
LITERAL AND CONTROL OPERATIONS								
ANDLW	k	AND literal with W	1	1110	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	1001	kkkk	kkkk	None	1
CLRWDT	k	Clear watchdog timer	1	0000	0000	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Unconditional branch	2	101k	kkkk	kkkk	None	
IORLW	k	Inclusive OR literal with W	1	1101	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	1100	kkkk	kkkk	None	
OPTION	k	Load OPTION register	1	0000	0000	0010	None	
RETLW	k	Return, place literal in W	2	1000	kkkk	kkkk	None	
SLEEP	-	Go into standby mode	1	0000	0000	0011	$\overline{TO}$,$\overline{PD}$	
TRIS	f	Load TRIS register	1	0000	0000	0fff	None	3
XORLW	k	Exclusive OR Literal to W	1	1111	kkkk	kkkk	Z	

Note 1: the 9th bit of the program counter will be forced to a '0' by any instruction that writes to the PC except for GOTO.

2: When an I/O register is modified as a function of itself (e.g. `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

3: The instruction `TRIS f`, where f = 5, 6, or 7 causes the contents of the W register to be written to the tristate latches of PORTA, B or C, respectively. A '1' forces the pin to a hi-impedance state and disables the output buffers.

4: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared (if assigned to TMR0).

Tabel 7/6.5.2-13: Samenvatting van de PIC16C5x instructieset.

Gebruik slechts de laagste 4 bits van de ID-lokaties en programmeer de hoogste 8 bits als "1"-en.

Instructieset

Inleiding

Elke PIC16C5x-instructie is een 12 bit woord dat is opgedeeld in een OPCODE (die de soort instructie specificeert) en één of meer OPERANDS die de operatie van de instructie verder specificeren. In tabel 7/6.5.2-13 is de samenvatting van de instructieset gegroepeerd in byte-georiënteerde, bitgeoriënteerde en letterlijke en besturingsoperaties.

Voor byte-georiënteerde instructies stelt "f" een file register-designator voor en "d" een bestemmings- (destination-) designator. De file register-designator wordt gebruikt om één van de 32 file-registers die door de instructie worden gebruikt te specificeren.

De destination-designator specificeert waar het resultaat van de operatie naar toe gaat. Als "d" "0" is, wordt het resultaat in het W-register geplaatst. Als "d" "1" is, gaat het resultaat naar het file-register dat in de instructie is gespecificeerd.

Voor bit-georiënteerde instructies is "b" een bit-field designator die het nummer van het door de operatie beïnvloede bit selecteert, terwijl "f" het nummer van de file aangeeft, waarin het bit zich bevindt.

Voor letterlijke en besturingsoperaties, geeft "k" een 8 of 9 bit constante of een letterlijke waarde aan.

Alle instructies worden binnen één enkele instructiecyclus uitgevoerd. tenzij een conditionele test "waar" is of de program-counter als gevolg van een instructie werd veranderd. In dat geval zijn voor de uitvoering twee instructiecycli nodig. Een instructie bestaat uit vier oscillator-perioden. Bij een oscillatorfrequentie van 4 MHz is de normale uitvoeringstijd van een instructie dus 1 µs.

Overzicht instructies

Aangezien de instructieset maar 33 mogelijkheden biedt, worden de instructies hierna in de figuren 7/6.5.2-35a tot en met -35i allemaal uitgewerkt, om ze zo volledig mogelijk te verklaren. Tabel 7/6.5.2-14 geeft een overzicht van de hierbij gebruikte opcode field-beschrijvingen.

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0 (store result in W) d = 1 (store result in file register 'f') Default is d = 1
label	Label name
TOS	Top of Stack
PC	Program Counter
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-Out bit
$\overline{PD}$	Power-Down bit
dest	Destination, either the W register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

Tabel 7/6.5.2-14: Opcode field-beschrijvingen.

6.5 PIC16/17 typen 8 bit microcontrollers

ADDWF — **Add W and f**

Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (dest)
Status Affected:	C, DC, Z
Encoding:	0001 | 11df | ffff
Description:	Add the contents of the W register and register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example:	ADDWF FSR, 0

Before Instruction
W = 0x17
FSR = 0xC2,

After Instruction
W = 0xD9
FSR = 0xC2

ANDWF — **AND W with f**

Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (dest)
Status Affected:	Z
Encoding:	0001 | 01df | ffff
Description:	The contents of the W register are AND'ed with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example:	ANDWF FSR, 1

Before Instruction
W = 0x17
FSR = 0xC2

After Instruction
W = 0x17
FSR = 0x02

ANDLW — **And literal with W**

Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Encoding:	1110 | kkkk | kkkk
Description:	The contents of the W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example:	ANDLW 0x5F

Before Instruction
W = 0xA3

After Instruction
W = 0x03

BCF — **Bit Clear f**

Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → (f<b>)
Status Affected:	None
Encoding:	0100 | bbbf | ffff
Description:	Bit 'b' in register 'f' is cleared.
Words:	1
Cycles:	1
Example:	BCF FLAG_REG, 7

Before Instruction
FLAG_REG = 0xC7

After Instruction
FLAG_REG = 0x47

Figuur 7/6.5.2-35a: Uitwerking van de instructies, deel 1.

6.5 PIC16/17 typen 8 bit microcontrollers

BSF	Bit Set f
Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Encoding:	0101 | bbbf | ffff
Description:	Bit 'b' in register 'f' is set.
Words:	1
Cycles:	1
Example:	BSF FLAG_REG, 7

Before Instruction
FLAG_REG= 0x0A

After Instruction
FLAG_REG= 0x8A

BTFSC	Bit Test f, Skip if Clear
Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Encoding:	0110 | bbbf | ffff
Description:	If bit 'b' in register 'f' is 0 then the next instruction is skipped. If bit 'b' is 0 then the next instruction fetched during the current instruction execution is discarded, and an NOP is executed instead, making this a 2 cycle instruction.
Words:	1
Cycles:	1(2)
Example:	HERE BTFSC FLAG,1 FALSE GOTO PROCESS_CODE TRUE • • •

Before Instruction
PC = address (HERE)

After Instruction
if FLAG<1> = 0,
PC = address (TRUE);
if FLAG<1> = 1,
PC = address (FALSE)

BTFSS	Bit Test f, Skip if Set
Syntax:	[*label*] BTFSS f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Encoding:	0111 | bbbf | ffff
Description:	If bit 'b' in register 'f' is '1' then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and an NOP is executed instead, making this a 2 cycle instruction.
Words:	1
Cycles:	1(2)
Example:	HERE BTFSS FLAG,1 FALSE GOTO PROCESS_CODE TRUE • • •

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0,
PC = address (FALSE);
if FLAG<1> = 1,
PC = address (TRUE)

Figuur 7/6.5.2-35b: Uitwerking van de instructies, deel 2.

6.5 PIC16/17 typen 8 bit microcontrollers

CALL — **Subroutine Call**

Syntax: [*label*] CALL k

Operands: $0 \leq k \leq 2047$

Operation: (PC) + 1→ Top of Stack;
k → PC<8:0>;
(STATUS<6:5>) → PC<10:9>;
0 → PC<8>

Status Affected: None

Encoding:

1001	kkkk	kkkk

Description: Subroutine call. First, return address (PC+1) is pushed onto the stack. The eight bit immediate address is loaded into PC bits <7:0>. The upper bits PC<10:9> are loaded from STATUS<6:5>, PC<8> is cleared. CALL is a two cycle instruction.

Words: 1

Cycles: 2

Example: HERE CALL THERE

Before Instruction
PC = address (HERE)

After Instruction
PC = address (THERE)
TOS = address (HERE)

CLRF — **Clear f**

Syntax: [*label*] CLRF f

Operands: $0 \leq f \leq 127$

Operation: 00h → (f);
1 → Z

Status Affected: Z

Encoding:

0000	011f	ffff

Description: The contents of register 'f' are cleared and the Z bit is set.

Words: 1

Cycles: 1

Example: CLRF FLAG_REG

Before Instruction
FLAG_REG = 0x5A

After Instruction
FLAG_REG = 0x00
Z = 1

CLRW — **Clear W**

Syntax: [*label*] CLRW

Operands: None

Operation: 00h → (W);
1 → Z

Status Affected: Z

Encoding:

0000	0100	0000

Description: The W register is cleared. Zero bit (Z) is set.

Words: 1

Cycles: 1

Example: CLRW

Before Instruction
W = 0x5A

After Instruction
W = 0x00
Z = 1

CLRWDT — **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT;
0 → WDT prescaler;
1 → $\overline{TO}$;
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0100

Description: The CLRWDT instruction resets the WDT. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Example: CLRWDT

Before Instruction
WDT counter = ?

After Instruction
WDT counter = 0x00
WDT prescale = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

Figuur 7/6.5.2-35c: Uitwerking van de instructies, deel 3.

6.5 PIC16/17 typen 8 bit microcontrollers

COMF — Complement f

Syntax: [*label*] COMF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: ($\bar{f}$) → (dest)

Status Affected: Z

Encoding:

0010	01df	ffff

Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example: `COMF REG1,0`

Before Instruction
REG1 = 0x13

After Instruction
REG1 = 0x13
W = 0xEC

DECFSZ — Decrement f, Skip if 0

Syntax: [*label*] DECFSZ f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) – 1 → d; skip if result = 0

Status Affected: None

Encoding:

0010	11df	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, the next instruction, which is already fetched, is discarded and an NOP is executed instead making it a two cycle instruction.

Words: 1

Cycles: 1(2)

Example:

```
HERE      DECFSZ   CNT, 1
          GOTO     LOOP
CONTINUE  •
          •
          •
```

Before Instruction
PC = address (HERE)

After Instruction
CNT = CNT - 1;
if CNT = 0,
PC = address (CONTINUE);
if CNT ≠ 0,
PC = address (HERE+1)

DECF — Decrement f

Syntax: [*label*] DECF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) – 1 → (dest)

Status Affected: Z

Encoding:

0000	11df	ffff

Description: Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example: `DECF CNT, 1`

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

GOTO — Unconditional Branch

Syntax: [*label*] GOTO k

Operands: 0 ≤ k ≤ 2047

Operation: k → PC<8:0>;
STATUS<6:5> → PC<10:9>

Status Affected: None

Encoding:

101k	kkkk	kkkk

Description: GOTO is an unconditional branch. The 9-bit immediate value is loaded into PC bits <8:0>. The upper bits of PC are loaded from STATUS<6:5>. GOTO is a two cycle instruction.

Words: 1

Cycles: 2

Example: `GOTO THERE`

After Instruction
PC = address (THERE)

Figuur 7/6.5.2-35d: Uitwerking van de instructies, deel 4.

6.5 PIC16/17 typen 8 bit microcontrollers

INCF — **Increment f**

Syntax: [*label*] INCF f,d

Operands: $0 \leq f \leq 127$
$d \in [0,1]$

Operation: (f) + 1 → (dest)

Status Affected: Z

Encoding: | 0010 | 10df | ffff |

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Example: `INCF CNT, 1`

Before Instruction
CNT = 0xFF
Z = 0

After Instruction
CNT = 0x00
Z = 1

INCFSZ — **Increment f, Skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: $0 \leq f \leq 127$
$d \in [0,1]$

Operation: (f) + 1 → (dest), skip if result = 0

Status Affected: None

Encoding: | 0011 | 11df | ffff |

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, then the next instruction, which is already fetched, is discarded and an NOP is executed instead making it a two cycle instruction.

Words: 1

Cycles: 1(2)

Example:

```
HERE      INCFSZ    CNT,  1
          GOTO      LOOP
CONTINUE  •
          •
          •
```

Before Instruction
PC = address (HERE)

After Instruction
CNT = CNT + 1;
if CNT = 0,
PC = address (CONTINUE);
if CNT ≠ 0,
PC = address (HERE +1)

IORLW — **Inclusive OR literal with W**

Syntax: [*label*] IORLW k

Operands: $0 \leq k \leq 255$

Operation: (W) .OR. (k) → (W)

Status Affected: Z

Encoding: | 1101 | kkkk | kkkk |

Description: The contents of the W register are OR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example: `IORLW 0x35`

Before Instruction
W = 0x9A

After Instruction
W = 0xBF

IORWF — **Inclusive OR W with f**

Syntax: [*label*] IORWF f,d

Operands: $0 \leq f \leq 127$
$d \in [0,1]$

Operation: (W) .OR. (f) → (dest)

Status Affected: Z

Encoding: | 0001 | 00df | ffff |

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Example: `IORWF RESULT, 0`

Before Instruction
RESULT = 0x13
W = 0x91

After Instruction
RESULT = 0x13
W = 0x93

Figuur 7/6.5.2-35e: Uitwerking van de instructies, deel 5.

6.5 PIC16/17 typen 8 bit microcontrollers

MOVF — **Move f**

Syntax:	[*label*] MOVF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) → (dest)
Status Affected:	Z
Encoding:	`0010` `00df` `ffff`
Description:	The contents of register 'f' is moved to destination 'd'. If 'd' is 0, destination is the W register. If 'd' is 1, the destination is file register 'f'. 'd' is 1 is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1
Example:	`MOVF FSR, 0`

After Instruction
W = value in FSR register

MOVLW — **Move Literal to W**

Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (W)
Status Affected:	None
Encoding:	`1100` `kkkk` `kkkk`
Description:	The eight bit literal 'k' is loaded into the W register. The don't cares will assemble as 0s.
Words:	1
Cycles:	1
Example:	`MOVLW 0x5A`

After Instruction
W = 0x5A

MOVWF — **Move W to f**

Syntax:	[*label*] MOVWF f
Operands:	0 ≤ f ≤ 127
Operation:	(W) → (f)
Status Affected:	None
Encoding:	`0000` `001f` `ffff`
Description:	Move data from the W register to register 'f'.
Words:	1
Cycles:	1
Example:	`MOVWF TEMP_REG`

Before Instruction
TEMP_REG = 0xFF
W = 0x4F

After Instruction
TEMP_REG = 0x4F
W = 0x4F

NOP — **No Operation**

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	`0000` `0000` `0000`
Description:	No operation.
Words:	1
Cycles:	1
Example:	`NOP`

Figuur 7/6.5.2-35f: Uitwerking van de instructies, deel 6.

6.5 PIC16/17 typen 8 bit microcontrollers

OPTION — **Load OPTION Register**

Syntax: [*label*] OPTION

Operands: None

Operation: (W) → OPTION

Status Affected: None

Encoding:

0000	0000	0010

Description: The content of the W register is loaded into the OPTION register.

Words: 1

Cycles: 1

Example OPTION

Before Instruction
W = 0x07

After Instruction
OPTION = 0x07

RETLW — **Return, place literal in W**

Syntax: [*label*] RETLW k

Operands: $0 \leq k \leq 255$

Operation: k → (W);
TOS → PC

Status Affected: None

Encoding:

1000	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.

Words: 1

Cycles: 2

Example:

```
        CALL TABLE  ;W contains
                    ;table
TABLE               ;offset value
        •           ;W now has table
                       value•
        •
        ADDWF PC    ;W = offset
        RETLW k1    ;Begin table
        RETLW k2    ;
        •
        •
        •
        RETLW kn    ; End of table
```

Before Instruction
W = 0x07

After Instruction
W = value of k7

RLF — **Rotate Left f through Carry**

Syntax: [*label*] RLF f,d

Operands: $0 \leq f \leq 127$
$d \in [0,1]$

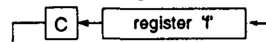
Operation: See description below

Status Affected: C

Encoding:

0011	01df	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.



Words: 1

Cycles: 1

Example: RLF REG1,0

Before Instruction
REG1 = 1110 0110
C = 0

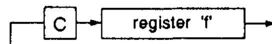
After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

Figuur 7/6.5.2-35g: Uitwerking van de instructies, deel 7.

6.5 PIC16/17 typen 8 bit microcontrollers

RRF **Rotate Right f through Carry**

Syntax:	[*label*] RRF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	See description below
Status Affected:	C
Encoding:	0011 | 00df | ffff
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words:	1
Cycles:	1
Example:	RRF REG1,0

Before Instruction
REG1 = 1110 0110
C = 0

After Instruction
REG1 = 1110 0110
W = 0111 0011
C = 1

SLEEP **Enter SLEEP Mode**

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT; 0 → WDT prescaler; 1 → $\overline{TO}$; 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$
Encoding:	0000 | 0000 | 0011
Description:	Time-out status bit ($\overline{TO}$) is set. The power down status bit ($\overline{PD}$) is cleared. The WDT and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. See section on SLEEP for more details.
Words:	1
Cycles:	1
Example:	SLEEP

SUBWF **Subtract W from f**

Syntax:	[*label*] SUBWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f) – (W) → (dest)
Status Affected:	C, DC, Z
Encoding:	0000 | 10df | ffff
Description:	Subtract (2's complement method) the W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example 1: SUBWF REG1, 1

Before Instruction
REG1 = 3
W = 2
C = ?

After Instruction
REG1 = 1
W = 2
C = 1 ; result is positive

Example 2:

Before Instruction
REG1 = 2
W = 2
C = ?

After Instruction
REG1 = 0
W = 2
C = 1 ; result is zero

Example 3:

Before Instruction
REG1 = 1
W = 2
C = ?

After Instruction
REG1 = FF
W = 2
C = 0 ; result is negative

Figuur 7/6.5.2-35h: Uitwerking van de instructies, deel 8.

6.5 PIC16/17 typen 8 bit microcontrollers

SWAPF — **Swap f**

Syntax:	[*label*] SWAPF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<3:0>) → (dest<7:4>); (f<7:4>) → (dest<3:0>)
Status Affected:	None
Encoding:	0011 | 10df | ffff
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.
Words:	1
Cycles:	1
Example	SWAPF REG1, 0

Before Instruction
REG1 = 0xA5

After Instruction
REG1 = 0xA5
W = 0X5A

TRIS — **Load TRIS Register**

Syntax:	[*label*] TRIS f
Operands:	5 ≤ f ≤ 7
Operation:	(W) → TRIS register f
Status Affected:	None
Encoding:	0000 | 0000 | 0fff
Description:	TRIS register 'f' (f = 5, 6, or 7) is loaded with the contents of the W register
Words:	1
Cycles:	1
Example	TRIS PORTA

Before Instruction
W = 0XA5

After Instruction
TRISA = 0XA5

XORLW — **Exclusive OR literal with W**

Syntax:	[*label*] XORLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .XOR. k → (W)
Status Affected:	Z
Encoding:	1111 | kkkk | kkkk
Description:	The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example:	XORLW 0xAF

Before Instruction
W = 0xB5

After Instruction
W = 0x1A

XORWF — **Exclusive OR W with f**

Syntax:	[*label*] XORWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .XOR. (f) → (dest)
Status Affected:	Z
Encoding:	0001 | 10df | ffff
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example	XORWF REG 1

Before Instruction
REG = 0xAF
W = 0xB5

After Instruction
REG = 0x1A
W = 0xB5

Figuur 7/6.5.2-35i: Uitwerking van de instructies, deel 9.

6.5 PIC16/17 typen 8 bit microcontrollers

Elektrische en timing karakteristieken

In de figuren 7/6.5-2.36 tot en met -40 en de tabellen 7/6.5.2-15 tot en met -23 wordt ten slotte een overzicht gegeven van de elektrische- en timing eigenschappen van de PIC16C5x-familie.

Ambient temperature under bias	-55°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	0 to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS}	0 to +14V
Voltage on all other pins with respect to V_{SS}	-0.6V to V_{DD} + 0.6V
Total power dissipation (Note 1)	800 mW
Max. current out of V_{SS} pin	150 mA
Max. current into V_{DD} pin	50 mA
Max. current into an input pin (T0CKI only)	±500 µA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD})	±20 mA
Output clamp current, I_{OK} (V_0 < 0 or V_0 > V_{DD})	±20 mA
Max. output current sunk by any I/O pin	25 mA
Max. output current sourced by any I/O pin	20 mA
Max. output current sourced by a single I/O port (PORTA, B or C)	40 mA
Max. output current sunk by a single I/O port (PORTA, B or C)	50 mA

Note 1: Total power dissipation should not exceed 800 mW for the package. Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$$

Tabel 7/6.5.2-15: Maximaal toegelaten waarden.

OSC	16C5X-04	16C5X-10	16C5X-20	16LC5X-04
RC	V_{DD}: 3.0V to 6.25V I_{DD}: 2.4 mA Max. at 5.5V I_{PD}: 4 µA Max. at 3.0V WDT dis Freq: 4 MHz Max.	V_{DD}: 4.5V to 5.5V I_{DD}: 1.7 mA typ. at 5.5V I_{PD}: 0.25 µA typ. at 3.0V WDT dis Freq: 4 MHz Max.	V_{DD}: 4.5V to 5.5V I_{DD}: 1.7 mA typ. at 5.5V I_{PD}: 0.25 µA typ. at 3.0V WDT dis Freq: 4 MHz Max.	V_{DD}: 2.5V to 6.25V I_{DD}: 0.5 mA typ. at 5.5V I_{PD}: 4 µA Max. at 2.5V WDT dis Freq: 2 MHz Max.
XT	V_{DD}: 3.0V to 6.25V I_{DD}: 2.4 mA Max. at 5.5V I_{PD}: 5 µA Max. at 3.0V WDT dis Freq: 4 MHz Max.	V_{DD}: 4.5V to 5.5V I_{DD}: 1.7 mA typ. at 5.5V I_{PD}: 0.25 µA typ. at 3.0V WDT dis Freq: 4 MHz Max.	V_{DD}: 4.5V to 5.5V I_{DD}: 1.7 mA typ. at 5.5V I_{PD}: 0.25 µA typ. at 3.0V WDT dis Freq: 4 MHz Max.	V_{DD}: 2.5V to 6.25V I_{DD} 0.5 mA typ. at 5.5V I_{PD}: 4 µA Max. at 2.5V WDT dis Freq: 4 MHz Max.
HS	V_{DD}: 4.5V to 5.5V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 0.3 µA typ. at 3.0V WDT dis Freq: 4 MHz Max.	V_{DD}: 4.5V to 5.5V I_{DD}: 8 mA Max. at 5.5V I_{PD}: 4 µA Max. at 3.0V WDT dis Freq: 10 MHz Max.	V_{DD}: 4.5V to 5.5V I_{DD}: 16 mA Max. at 5.5V I_{PD}: 4 µA Max. at 3.0V WDT dis Freq: 20 MHz Max.	Do not use in HS mode
LP	V_{DD}: 4.5V to 5.5V I_{DD}: 17 µA typ. at 32 kHz, 3.0V I_{PD}: 0.3 µA typ. at 3.0V WDT dis Freq: 200 kHz typ.	Do not use in LP mode	Do not use in LP mode	V_{DD}: 2.5V to 6.25V I_{DD}: 11 µA typ. at 32 kHz, 2.5V I_{PD}: 0.25 µA typ. at 2.5V WDT dis Freq: 200 kHz typ.

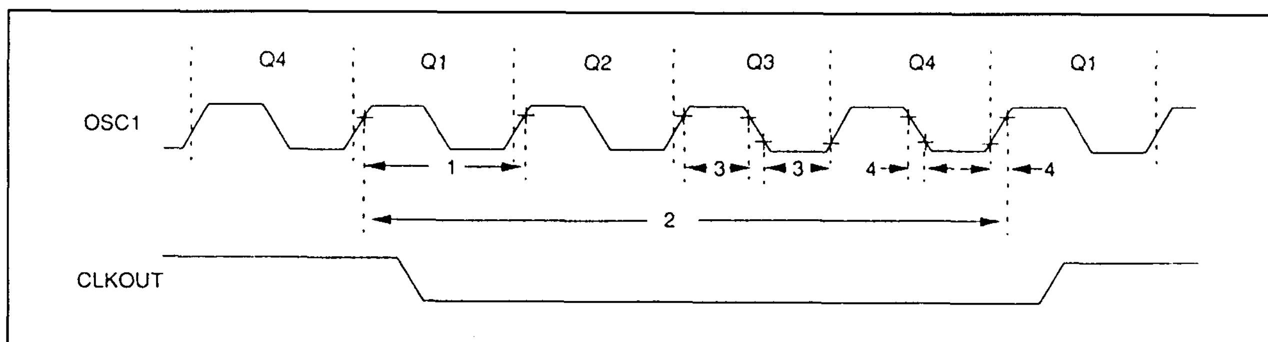
Tabel 7/6.5.2-16: Kort overzicht van de specificaties voor diverse oscillator-configuraties en bedrijfsfrequenties (commerciële typen).

6.5 PIC16/17 typen 8 bit microcontrollers

DC Characteristics Power Supply Pins	Standard Operating Conditions Operating Temperature 0°C ≤ TA ≤ +70°C, unless otherwise stated. Operating Voltage VDD = 3.0V to 5.5V, unless otherwise stated.					
Characteristic	**Sym**	**Min**	**Typ (Note 1)**	**Max**	**Units**	**Conditions**
Supply Voltage						
PIC16C5X-XT	VDD	3.0		6.25	V	FOSC = DC to 4 MHz
PIC16C5X-RC		3.0		6.25	V	FOSC = DC to 4 MHz
PIC16C5X-HS		4.5		5.5	V	FOSC = DC to 20 MHz
PIC16C5X-LP		2.5		6.25	V	FOSC = DC to 40 kHz
RAM Data Retention Voltage (Note 3)	VDR		1.5		V	Device in SLEEP mode
VDD start voltage to guarantee Power-On Reset	VPOR		VSS		V	See Section 7.4 for details on Power-On Reset
VDD rise rate to guarantee Power-On Reset	SVDD	0.05*			V/ms	See Section 7.4 for details on Power-On Reset
Supply Current (Note 2)						
PIC16C5X-XT	IDD		1.8	3.3	mA	FOSC = 4 MHz, VDD = 5.5V
PIC16C5X-RC (Note 5)			1.8	3.3	mA	FOSC = 4 MHz, VDD = 5.5V
PIC16C5X-HS			4.8	10	mA	FOSC = 10 MHz, VDD = 5.5V
			9.0	20	mA	FOSC = 20 MHz, VDD = 5.5V
PIC16C5X-LP			15	32	µA	FOSC = 32 kHz, VDD = 3.0V, WDT disabled
Power Down Current (Note 4)						
PIC16C5X	IPD		4	12	µA	VDD = 3.0V, WDT enabled
			0.6	9	µA	VDD = 3.0V, WDT disabled

* These parameters are based on characterization and are not tested.

Tabel 7/6.5.2-17: Gelijkspanningskenmerken van commerciële typen (alleen de voedings-aansluitingen).



Figuur 7/6.5.2-36: Externe clock-timing.

6.5 PIC16/17 typen 8 bit microcontrollers

DC Characteristics All Pins Except Power Supply Pins	Standard Operating Conditions Operating Temperature -40°C ≤ T_A ≤ +85°C (for industrial) 0°C ≤ T_A ≤ +70°C (for commercial) Operating Voltage V_{DD} range is described in Section 9.1 and Section 9.2.					
Characteristic	**Sym**	**Min**	**Typ (Note 1)**	**Max**	**Units**	**Conditions**
Input Low Voltage						
I/O ports	V_{IL}	V_{SS}		0.2 V_{DD}	V	Pin at hi-impedance
$\overline{MCLR}$ (Schmitt Trigger)		V_{SS}		0.15 V_{DD}	V	
TOCKI (Schmitt Trigger)		V_{SS}		0.15 V_{DD}	V	
OSC1 (Schmitt Trigger)		V_{SS}		0.15 V_{DD}	V	PIC16C5X-RC only (Note 5)
OSC1		V_{SS}		0.3 V_{DD}	V	PIC16C5X-XT, HS, LP
Input High Voltage						
I/O ports	V_{IH}	0.45 V_{DD}		V_{DD}	V	For all V_{DD} (Note 6)
		2.0		V_{DD}	V	4.0V < V_{DD} ≤ 5.5V (Note 6)
		0.36 V_{DD}		V_{DD}	V	V_{DD} > 5.5V
$\overline{MCLR}$ (Schmitt Trigger)		0.85 V_{DD}		V_{DD}	V	
TOCKI (Schmitt Trigger)		0.85 V_{DD}		V_{DD}	V	
OSC1 (Schmitt Trigger)		0.85 V_{DD}		V_{DD}	V	PIC16C5X-RC only (Note 5)
OSC1		0.7 V_{DD}		V_{DD}	V	PIC16C5X-XT, HS, LP
Input Leakage Current (Notes 3,4)						**For V_{DD} ≤ 5.5V**
I/O ports	I_{IL}	-1	0.5	+1	µA	V_{SS} ≤ V_{PIN} ≤ V_{DD}, Pin at hi-impedance
$\overline{MCLR}$		-5			µA	V_{PIN} = V_{SS} + 0.25V
$\overline{MCLR}$			0.5	+5	µA	V_{PIN} = V_{DD}
TOCKI		-3	0.5	+3	µA	V_{SS} ≤ V_{PIN} ≤ V_{DD}
OSC1		-3	0.5	+3	µA	V_{SS} ≤ V_{PIN} ≤ V_{DD} PIC16C5X-XT, HS, LP
Output Low Voltage						
I/O ports	V_{OL}			0.6	V	I_{OL} = 8.7 mA, V_{DD} = 4.5V
OSC2/CLKOUT (PIC16C5X-RC)				0.6	V	I_{OL} = 1.6 mA, V_{DD} = 4.5V
Output High Voltage						
I/O ports (Note 4)	V_{OH}	V_{DD}-0.7			V	I_{OH} = -5.4 mA, V_{DD} = 4.5V
OSC2/CLKOUT (PIC16C5X-RC)		V_{DD}-0.7			V	I_{OH} = -1.0 mA, V_{DD} = 4.5V

Note 1: Data in the column labeled "Typical" is based on characterization results at 25°C. This data is for design guidance only and is not tested for, or guaranteed by Microchip Technology.
2: Total power dissipation as stated under absolute maximum ratings must not be exceeded.
3: The leakage current on the $\overline{MCLR}$/V_{PP} pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltage.
4: Negative current is defined as coming out of the pin.
5: For PIC16C5X-RC devices, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC16C5X be driven with external clock in RC mode.
6: The user may use the better of the two specifications.

Tabel 7/6.5.2-18: Gelijkspanningskenmerken van commerciële typen (de overige aansluitpennen).

6.5 PIC16/17 typen 8 bit microcontrollers

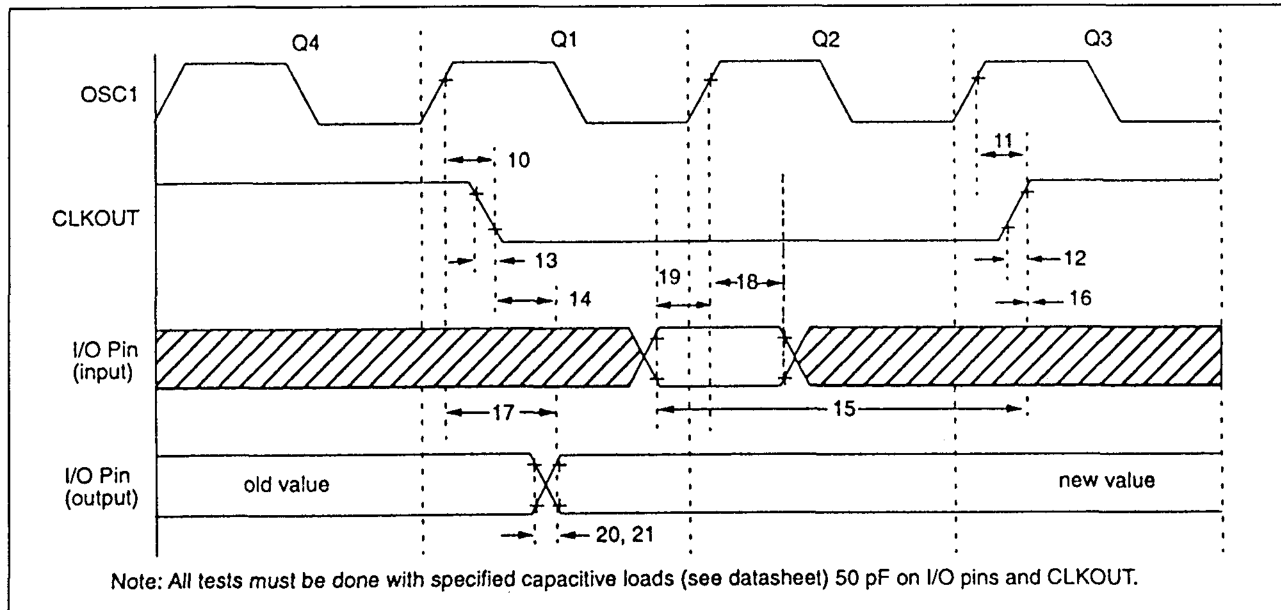
Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	External CLKIN Frequency (Note 1)	DC	—	4	MHz	RC osc mode
			DC	—	4	MHz	XT osc mode
			DC	—	20	MHz	HS osc mode (Comm/Indust)
			DC	—	16	MHz	HS osc mode (Automotive)
			DC	—	40	kHz	LP osc mode
		Oscillator Frequency (Note 1)	DC	—	4	MHz	RC osc mode
			0.1	—	4	MHz	XT osc mode
			4	—	20	MHz	HS osc mode (Comm/Indust)
			4	—	16	MHz	HS osc mode (Automotive)
			5	—	40	kHz	LP osc mode
1	Tosc	External CLKIN Period (Note 1)	250	—	—	ns	RC osc mode
			250	—	—	ns	XT osc mode
			50	—	—	ns	HS osc mode
			100	—	—	µs	LP osc mode
		Oscillator Period (Note 1)	250	—	—	ns	RC osc mode
			250	—	10,000	ns	XT osc mode
			62.5	—	250	ns	HS osc mode (Comm/Indust)
			50	—	250	ns	HS osc mode (Automotive)
			100	—	200	µs	LP osc mode
2	TCY	Instruction Cycle Time (Note 1)	1.0	—	DC	µs	
3	TosL, TosH	Clock in (OSC1) Low or High Time	50	—	—	ns	XT oscillator
			2.5	—	—	µs	LP oscillator
			10	—	—	ns	HS oscillator
4	TosR, TosF	Clock in (OSC1) Rise or Fall Time	25	—	—	ns	XT oscillator
			50	—	—	ns	LP oscillator
			15	—	—	ns	HS oscillator

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1/CLKIN pin.
When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

Tabel 7/6.5.2-19: Eisen die aan de externe clock worden gesteld (zie ook figuur 7/6.5.2-36).

6.5 PIC16/17 typen 8 bit microcontrollers



Figuur 7/6.5.2-37: CLKOUT en I/O-timing.

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1↑ to CLKOUT↓	—	15	30	ns	Note 1
11	TosH2ckH	OSC1↑ to CLKOUT↑	—	15	30	ns	Note 1
12	TckR	CLKOUT rise time	—	5	15	ns	Note 1
13	TckF	CLKOUT fall time	—	5	15	ns	Note 1
14	TckL2ioV	CLKOUT↓ to Port out valid	—	—	0.5 TCY+20	ns	Note 1
15	TioV2ckH	Port in valid before CLKOUT↑	0.25 TCY+25	—	—	ns	Note 1
16	TckH2ioI	Port in hold after CLKOUT↑	0	—	—	ns	Note 1
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	—	—	80 - 100	ns	Note 2
18	TosH2ioI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	TBD	—	—	ns	
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)	TBD	—	—	ns	
20	TioR	Port output rise time	—	10	25	ns	Note 2
21	TioF	Port output fall time	—	10	25	ns	Note 2

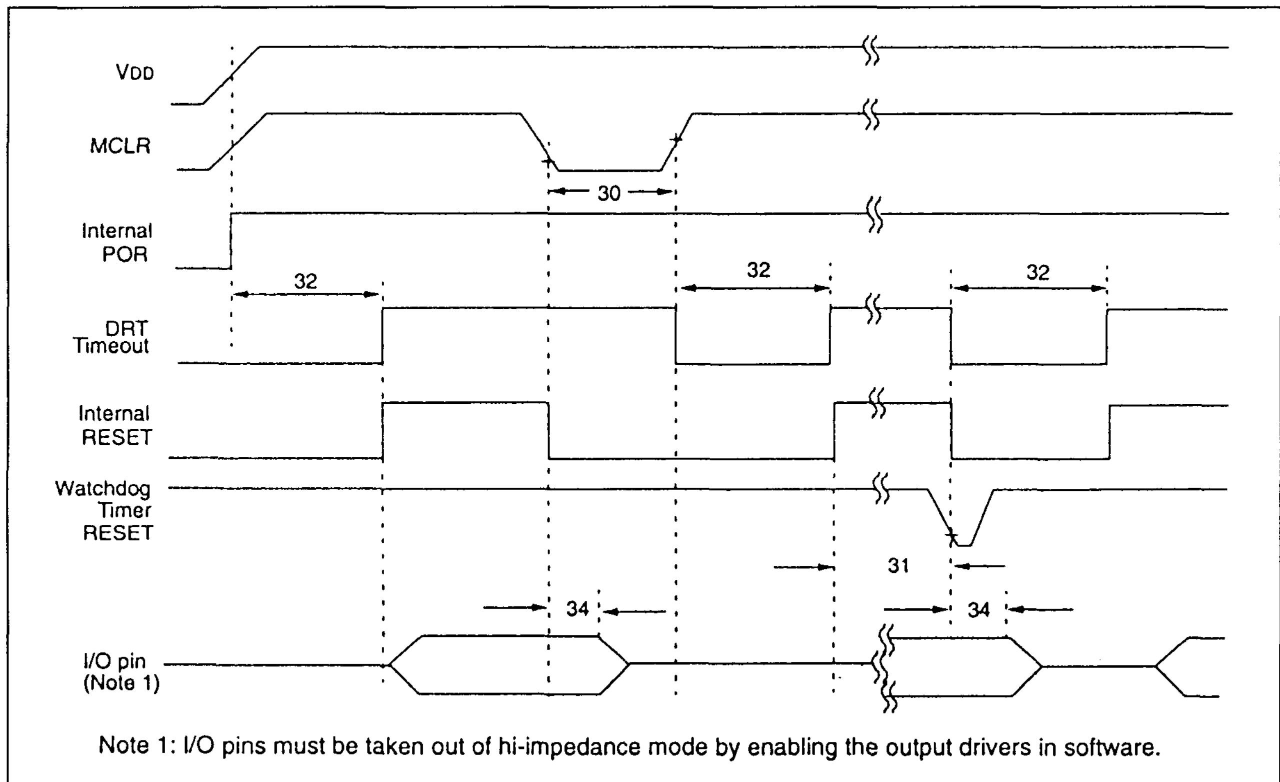
* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Measurements are taken in RC Mode where CLKOUT output is 4 x Tosc.

Tabel 7/6.5.2-20: Eisen die aan het CLKOUT-signaal en I/O-timing worden gesteld (zie ook figuur 7/6.5.2-37).

6.5 PIC16/17 typen 8 bit microcontrollers



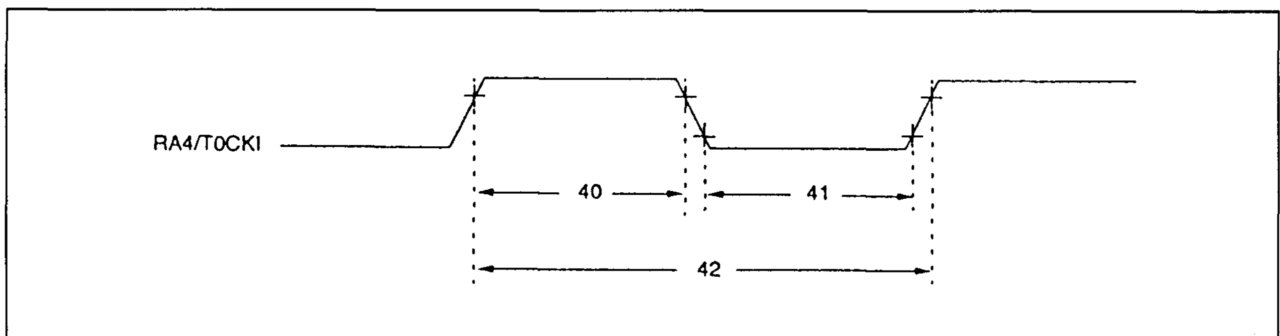
Figuur 7/6.5.2-38: Reset, Watchdog Timer en Device Reset Timer timing.

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	TmcL	$\overline{MCLR}$ Pulse Width (low)	100	—	—	ns	VDD = 5V, -40°C to +125°C
31	Twdt	Watchdog Timer Timeout Period (No Prescaler)	9*	18	30*	ms	VDD = 5V, -40°C to +125°C
32	TDRT	Device Reset Timer Period	9*	18*	30*	ms	VDD = 5V, -40°C to +125°C
34	Tioz	I/O Hi-impedance from $\overline{MCLR}$ Low			100	ns	

* These parameters are characterized but not tested.

† *Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only* and are not tested.

Tabel 7/6.5.2-21: De timing van Reset, Watchdog Timer en Device Reset Timer (zie ook figuur 7/6.5.2-38).



Figuur 7/6.5.2-39: TIMER0 clock-timing.

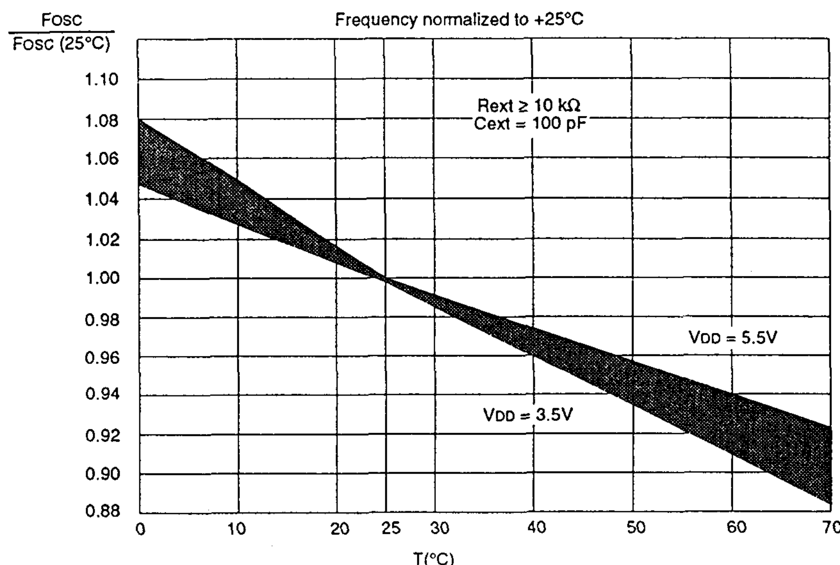
6.5 PIC16/17 typen 8 bit microcontrollers

Parameter No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5 TCY + 20*	—	—	ns	
			With Prescaler	10*	—	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5 TCY + 20*	—	—	ns	
			With Prescaler	10*	—	—	ns	
42	Tt0P	T0CKI Period		(TCY + 40*) / N	—	—	ns	N = prescale value (1, 2, 4, ..., 256)

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Tabel 7/6.5.2-22: Eisen die aan de TIMER0-clock worden gesteld (zie ook figuur 7/6.5.2-37).



Figuur 7/6.5.2-40: Typische RC-oscillator frequentie als functie van de temperatuur.

Cext	Rext	Average Fosc @ 5V, 25°C	
20 pF	3.3k	4.973 MHz	± 27%
	5k	3.82 MHz	± 21%
	10k	2.22 MHz	± 21%
	100k	262.15 kHz	± 31%
100 pF	3.3k	1.63 MHz	± 13%
	5k	1.19 MHz	± 13%
	10k	684.64 kHz	± 18%
	100k	71.56 kHz	± 25%
300 pF	3.3k	660.0 kHz	± 10%
	5.k	484.1 kHz	± 14%
	10k	267.63 kHz	± 15%
	160k	29.44 kHz	± 19%

The frequencies are measured on DIP packages.

The percentage variation indicated here is part to part variation due to normal process distribution. The variation indicated is ±3 standard deviation from average value for VDD = 5V.

Tabel 7/6.5.2-23: Enkele voorbeelden van RC-oscillator frequenties.

7/6.5.3

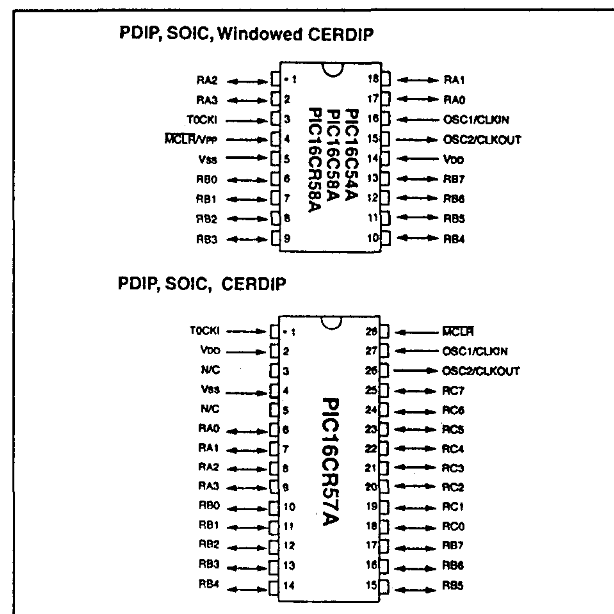
Type-beschrijving ENHANCED PIC16C5x-typen

Inleiding

De ***Enhanced*** (verbeterde) PIC16C5x-familie van Microchip Technology is een reeks goedkope, 8 bit, volledig statische, op EPROM gebaseerde CMOS microcontrollers. Deze familie (met de toevoeging "A" aan de codenaam) is natuurlijk pin- en software-compatibel met de gewone PIC16C5x-familie. Ook hier wordt gebruik gemaakt van een RISC-achtige architectuur met 33 single word/single cycle instructies. Voor alle instructies is slechts één cyclus (200 ns) nodig, behalve voor branch-instructies die er twee nodig hebben. De 12 bit brede instructies zijn in hoge mate symmetrisch, zodat een 2:1 code compressie ten opzichte van andere 8 bit microcontrollers mogelijk is. Ook de Enhanced PIC16C5x produkten zijn voorzien van speciale kenmerken om de systeemkosten en de eisen die aan de voeding worden gesteld te beperken, zoals de Power-On Reset (POR) en de Device Reset Timer. Er zijn vier oscillator-configuraties, inclusief de LP (Low Power) en de goedkope RC-oscillator.

Device	Pins	I/O	EPROM/ ROM	RAM
PIC16C54A	18	12	512	25
PIC16C58A	18	12	2K	73
PIC16CR58A	18	12	2K	73
PIC16CR57A	28	20	2K	72

Figuur 7/6.5.3-1: Overzicht van de Enhanced PIC16C5x-typen.



Figuur 7/6.5.3-2: Aansluitingen van de Enhanced PIC16C5x-typen, in PDIP-, SOIC- en CERDIP-uitvoering (met venster).

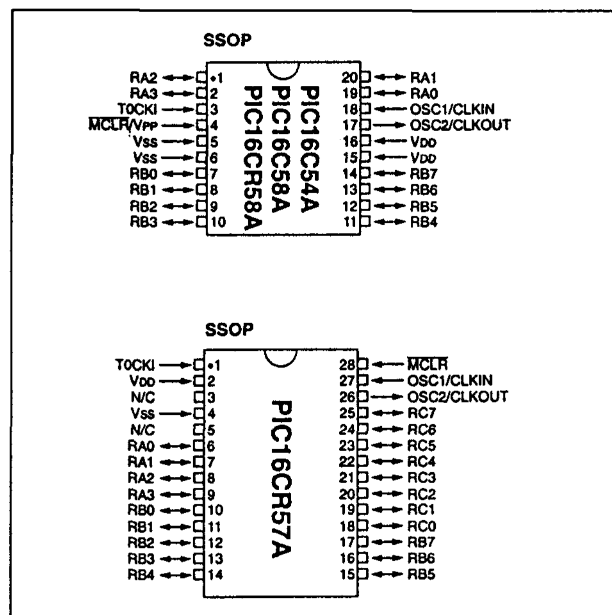
Verder kan gebruik worden gemaakt van de SLEEP mode, de watchdog timer en code beveiligingen. Met de UV-wisbare CERDIP-versies kan de code worden ontwikkeld, terwijl de OTP-versies geschikt zijn voor produktie-aantallen. Hierna worden van de Enhanced typen alleen de specifieke gegevens behandeld en de kenmerken die afwijken van de gewone typen.

Algemene gegevens

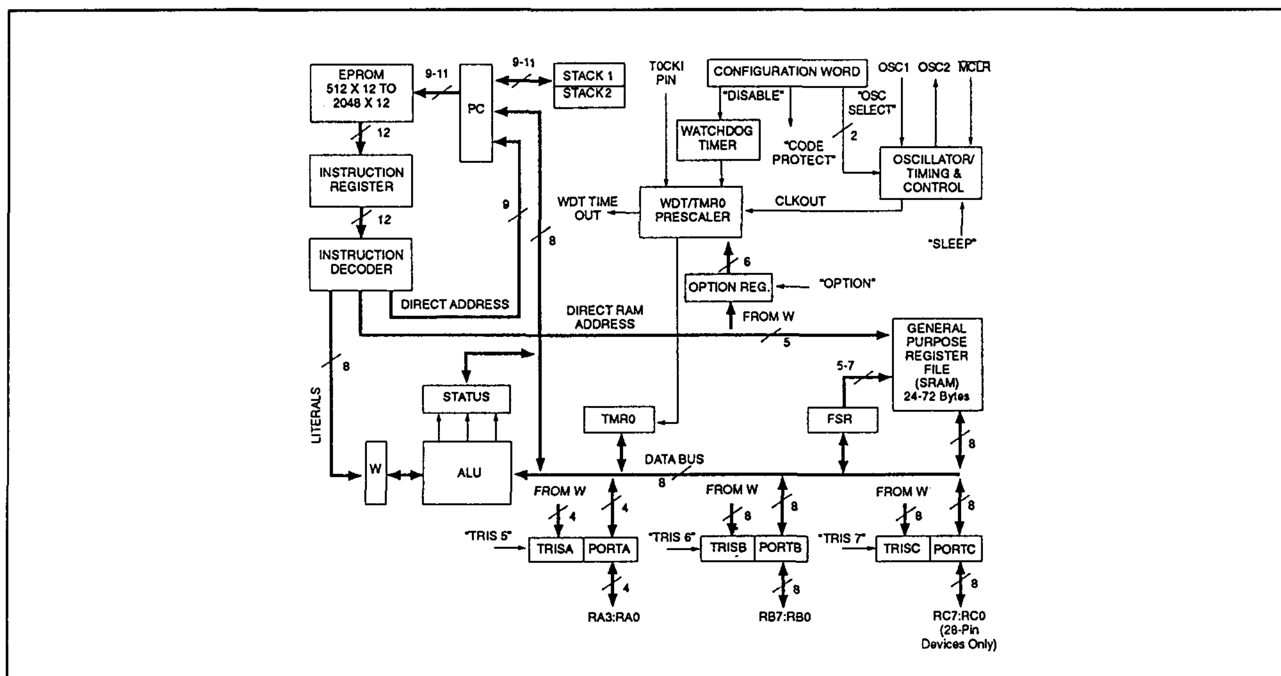
- behandelde typen:
  PIC16C54A, PIC16C58A, PIC16CR58A, PIC16CR57A (zie ook figuur 7/6.5.3-1)

6.5 PIC16/17 typen 8 bit microcontrollers

- 33 single-word instructies
- single-cycle instructies (200 ns) behalve branches
- snelheid:
  DC - 20 MHz clock input
  DC - 200 ns instructie-cyclus
- 12 bit brede instructies, 8 bit brede data
- 7 of 8 special function hardware registers
- hardware stack: twee niveaus diep
- directe, indirecte en relatieve adresseer-modes voor data en instructies
- 8 bit real-time clock/counter (TMR0) met 8 bit programmeerbare prescaler
- power-on reset (POR) en device reset timer
- watchdog timer (WDT) met eigen on-chip RC-oscillator
- programmeerbare code-beveiliging
- energiebesparende SLEEP mode
- selecteerbare oscillator-opties: RC, XT, HS en LP
- volledig statisch CMOS-ontwerp
- voedingsspanningen:
  EPROM-typen: 2,5 V - 6,25 V
  ROM-type: 2 V - 6,25 V
- gering energie-verbruik:
  <2 mA typ. (5 V, 4 MHz)
  15 µA typ. (3 V, 32 kHz)
  <0,3µA typ. (standby)
- fabrikant: Microchip Technology



Figuur 7/6.5.3-3: Aansluitingen van de Enhanced PIC16C5x-typen, in SSOP-uitvoering.



Figuur 7/6.5.3-4: Vereenvoudigd blokschema van de PIC16C5x-schakelingen.

6.5 PIC16/17 typen 8 bit microcontrollers

Name	DIP, SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
RA0 RA1 RA2 RA3	17 18 1 2	19 20 1 2	I/O I/O I/O I/O	TTL TTL TTL TTL	Bi-directional I/O port
RB0 RB1 RB2 RB3 RB4 RB5 RB6 RB7	6 7 8 9 10 11 12 13	7 8 9 10 11 12 13 14	I/O I/O I/O I/O I/O I/O I/O I/O	TTL TTL TTL TTL TTL TTL TTL TTL	Bi-directional I/O port
T0CKI	3	3	I	ST	Clock input to TMR0 timer. Must be tied to VSS or VDD, if not in use, to reduce current consumption.
$\overline{MCLR}$/VPP	4	4	I	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device. Voltage on the $\overline{MCLR}$/VPP pin must not exceed VDD to avoid unintended entering of test modes.
OSC1/CLKIN	16	18	I	ST	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	17	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
VDD	14	15,16	P	—	Positive supply for logic and I/O pins.
VSS	5	5,6	P	—	Ground reference for logic and I/O pins.

Legend: I = Input only; O = Output only; I/O = Input/Output; P = Power; — = Not Used; TTL = TTL input; ST = Schmitt Trigger input.

Figuur 7/6.5.3-5: Aansluitingen en signaalfuncties van de PIC16C54A, PIC16C58A en PIC16CR58A.

Architectuur

De Enhanced PIC16C5x-familie berust ook op de Harvard architectuur waarbij toegang tot programma en data via aparte bussen plaats vindt. De PIC16C58A/CR58A en PIC16CR57A adresseren 2 kB x 12 programma-geheugen, terwijl de PIC16C54A 512 B x 12 adresseert. Alle programma-geheugen is intern.

Vanwege de gelijkenissen wordt hierbij verwezen naar de hiervoor beschreven architectuur en werking van de PIC16C5x-familie.

- Architectuur:
  Zie uitleg registers, ALU, statusbits, enz., Clock-schema/Instructie-cyclus (figuur 7/6.5.2-7) en Instructie-afhandeling/pijplijnen (figuur 7/6.5.2-8).
- Geheugen-organisatie:
  Zie Programma-geheugen (figuur 7/6.5.2-9), Data-geheugen (figuur 7/6.5.2-10 en tabel 7/6.5.2-1), Status Register (figuur 7/6.5.2-11), Option Register (figuur 7/6.5.2-12), Indirecte data adressering, INDF en FSR Registers (figuur 7/6.5.2-13). Program Counter (tabel 7/6.5.2-2), W(erk) Register (tabellen 7/6.5.2-3 en -4).
- I/O-poorten:
  Zie PORTA (4-bit I/O-register). Alleen de laagste 4 bits worden gebruikt (RA3:RA0). De bits 7-4 zijn niet geïmplementeerd en worden als “0” uitgelezen; PORTB (8-bit I/O-register (PORTB:0); PORTC (PICR57A: 8-bit I/O-register, PIC16C54A/C58A/CR58A: General Purpose Register), TRIS Registers, I/O Interfacing (figuur 7/6.5.2-14), I/O Programmeer-overwegingen (figuur 7/6.5.2-15).

6.5 PIC16/17 typen 8 bit microcontrollers

Name	DIP, SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
RA0	6	5	I/O	TTL	Bi-directional I/O port
RA1	7	6	I/O	TTL	
RA2	8	7	I/O	TTL	
RA3	9	8	I/O	TTL	
RB0	10	9	I/O	TTL	Bi-directional I/O port
RB1	11	10	I/O	TTL	
RB2	12	11	I/O	TTL	
RB3	13	12	I/O	TTL	
RB4	14	13	I/O	TTL	
RB5	15	15	I/O	TTL	
RB6	16	16	I/O	TTL	
RB7	17	17	I/O	TTL	
RC0	18	18	I/O	TTL	Bi-directional I/O port
RC1	19	19	I/O	TTL	
RC2	20	20	I/O	TTL	
RC3	21	21	I/O	TTL	
RC4	22	22	I/O	TTL	
RC5	23	23	I/O	TTL	
RC6	24	24	I/O	TTL	
RC7	25	25	I/O	TTL	
TOCKI	1	2	I	ST	Clock input to TMR0 register. Must be tied to V_{SS} or V_{DD} if not in use to reduce current consumption.
$\overline{MCLR}$	28	28	I	ST	Master clear (reset) input. This pin is an active low reset to the device. Voltage on the $\overline{MCLR}$ pin must not exceed V_{DD} to avoid unintended entering of test modes.
OSC1/CLKIN	27	27	I	ST	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	26	26	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
V_{DD}	2	3,4	P	—	Positive supply for logic and I/O pins.
V_{SS}	4	1,14	P	—	Ground reference for logic and I/O pins.
N/C	3,5	—	—	—	Unused, do not connect

Legend: I = Input only; O = Output only; I/O = Input/Output; P = Power; — = Not Used; TTL = TTL input;
ST = Schmitt Trigger input.

Figuur 7/6.5.3-6: Aansluitingen en signaalfuncties van de PIC16CR57A.

- TIMER0 (TMR0) Module:
  Zie de figuren 7/6.5.2-16, -17, -18 en -19), Gebruik van TMR0 met een externe clock (figuur 7/6.5.2-20), Prescaler (figuur 7/6.5.2-21 en de tabellen 7/6.5.2-5 en -6).
- Speciale mogelijkheden van de CPU:
  Zie Configuratiebits (figuur 7/6.5.2-22), Oscillator-configuraties (LP: Low Power Crystal, XT: Crystal/Resonator, HS: High Speed Crystal/Resonator en RC: Resistor/Capacitor) (figuren 7/6.5.2-23 t/m -27 en de tabellen 7/6.5.2-7 en -8), Reset (Power-On Reset (POR), $\overline{MCLR}$ reset tijdens normaal bedrijf, $\overline{MCLR}$ reset tijdens SLEEP en WDT time-out reset) (tabellen 7/6.5.2-9, -10 en -11 en figuur 7/6.5.2-28), Power-On Reset (POR) en Device Reset Timer (DRT) (figuren 7/6.5.2-29 t/m -33), Watchdog Timer (WDT) (tabel 7/6.5.2-12 en figuur 7/6.5.2-34), Power-Down Mode (SLEEP), Code-beveiliging, ID-lokaties.

Instructieset

Ook de instructieset van de Enhanced PIC16C5x-familie is dezelfde als van de gewone.

6.5 PIC16/17 typen 8 bit microcontrollers

Ambient temperature under bias .... -55°C to +125°C
Storage temperature .... - 65°C to +150°C
Voltage on VDD with respect to Vss .... 0 to +7.5V
Voltage on $\overline{MCLR}$ with respect to Vss .... 0 to +14V
Voltage on all other pins with respect to Vss .... -0.6V to (VDD + 0.6V)
Total power dissipation (Note 1) .... 800 mW
Max. current out of Vss pin .... 150 mA
Max. current into VDD pin .... 100 mA
Max. current into an input pin (T0CKI only) .... ±500 μA
Input clamp current, IIK (VI < 0 or VI > VDD) .... ±20 mA
Output clamp current, IOK (V0 < 0 or V0 > VDD) .... ±20 mA
Max. output current sunk by any I/O pin .... 25 mA
Max. output current sourced by any I/O pin .... 20 mA
Max. output current sourced by a single I/O port (PORTA or B) .... 40 mA
Max. output current sunk by a single I/O port (PORTA or B) .... 50 mA

Note 1: Total power dissipation should not exceed 800 mW for the package. Power dissipation is calculated as follows:

$P_{dis} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD}-V_{OH}) \times I_{OH}\} + \Sigma(V_{OL} \times I_{OL})$

Tabel 7/6.5.3-1: Maximaal toegelaten waarden voor de PIC16C54A.

OSC	16C54A-04	16C54A-10	16C54A-20	16LC54A-04
RC	VDD: 3.0V to 6.25V IDD: 2.4 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 3.0V to 6.25V IDD: 1.7 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 3.0V to 6.25V IDD: 1.7 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 2.5V to 6.25V IDD: 0.5 mA typ. at 5.5V IPD: 0.25 μA typ. at 2.5V WDT dis Freq: 2 MHz max.
XT	VDD: 3.0V to 6.25V IDD 2.4 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 3.0V to 6.25V IDD: 1.7 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 3.0V to 6.25V IDD: 1.7 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 2.5V to 6.25V IDD: 0.5 mA typ. at 5.5V IPD: 0.25 μA typ. at 2.5V WDT dis Freq: 2 MHz max.
HS	VDD: 4.5V to 5.5V IDD: 2.4 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 8 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 10 MHz max.	VDD: 4.5V to 5.5V IDD: 16 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 20 MHz max.	Do not use in HS mode
LP	VDD: 3.0V to 6.25V IDD: 14 μA typ. at 32kHz, 3.0V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 200 kHz max.	Do not use in LP mode	Do not use in LP mode	VDD: 2.5V to 6.25V IDD: 27 μA max. at 32kHz, 2.5V IPD: 4 μA max. at 2.5V WDT dis Freq: 200 kHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications. It is recommended that the user select the device type that guarantees the specifications required.

Tabel 7/6.5.3-2: Overzicht van de specificaties voor diverse oscillator-configuraties en bedrijfsfrequenties voor commerciële typen van de PIC16C54A.

Er wordt dus verwezen naar de tabellen 7/6.5.2-13 (Instructieset) en -14 (toelichting op de instructieset) en figuur 7/6.5.2-35.
Elke instructie bestaat uit een 12 bit woord dat is opgedeeld in een OPCODE (die de soort instructie specificeert) en één of meer OPERANDS die de operatie van de instructie verder specificeren. Voor byte-georiënteerde instructies stelt "f" een file register-designator voor en "d" een destination-designator (bestemming). De file register-designator dient om één van de 32 file-registers die door de instructie worden gebruikt te specificeren.

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial. Operating voltage: VDD = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Supply Voltage	VDD	3.0 4.5	—	6.25 5.5	V	XT, RC and LP osc configuration HS osc configuration
RAM Data Retention Voltage (Note 1)	VDR	1.5	—	—	V	Device in SLEEP mode
VDD start voltage to guarantee Power-On Reset	VPOR	—	Vss	—	V	See section on Power-On Reset for details
VDD rise rate to guarantee Power-On Reset	SVDD	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	IDD					XT and RC options (Note 4)
		—	1.8	2.4	mA	Fosc = 4 MHz, VDD = 5.5V
						LP osc option, Commercial
		—	14		µA	Fosc = 32 kHz, VDD = 3.0V, WDT disabled
						LP option, Industrial
			17		µA	Fosc = 32 kHz, VDD = 3.0V, WDT disabled
Power Down Current (Note 3)						
WDT enabled	IPD	—	4	12	µA	VDD = 3.0V, Commercial
		—	5	14	µA	VDD = 3.0V, Industrial
WDT disabled		—	0.25	4	µA	VDD = 3.0V, Commercial
		—	0.3	5	µA	VDD = 3.0V, Industrial

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as bus loading, oscillator type, bus rate, internal code execution pattern, and temperature also have an impact on the current consumption.

a) The test conditions for all IDD measurements in active operation mode are:
OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD, TOCKI = VDD, $\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.
b) For stand-by current measurements, the conditions are the same, except that the device is in SLEEP mode.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and Vss.

4: RC mode does not include current through Rext. The current through the resistor can be estimated by the formula:
IR = VDD/2Rext (mA) with Rext in kOhm.

Tabel 7/6.5.3-3: Gelijkspanningskenmerken van de PIC16C54A-04 (4 MHz-versie, alleen de voeding).

De destination-designator specificeert de lokatie waar het resultaat van de operatie naar toe gaat. Als "d" "0" is, wordt het resultaat in het W-register geplaatst. Als "d" "1" is, gaat het resultaat naar het file-register dat in de instructie is gespecificeerd. Voor bit-georiënteerde instructies is "b" een bit-field designator die het nummer van het door de operatie beïnvloede bit selecteert, terwijl "f" het nummer van de file aangeeft, waarin het bit zich bevindt. Voor letterlijke en besturings-operaties, geeft "k" een 8 of 9 bit constante of een letterlijke waarde aan.

Alle instructies worden binnen één enkele instructiecyclus uitgevoerd, tenzij een conditionele test "waar" is of de program-counter als gevolg van een instructie werd veranderd. In dat geval zijn voor de uitvoering twee instructiecycli nodig. Een instructie bestaat uit vier oscillator-perioden. Bij een oscillator-frequentie van 4 MHz is de normale uitvoeringstijd van een instructie dus 1 µs.

Elektrische en timing-eigenschappen

De Enhanced PIC16C5x-typen wijken vooral af van de gewone typen op het gebied van

opgenomen en afgegeven vermogen. Voor de timingen van de signalen wordt verwezen naar:

- **CLKOUT en I/O-timing**:
  Tabel 7/6.5.2-20 en figuur 7/6.5.2-37;
- **Reset, Watchdog Timer en Device Reset Timer timing**:
  Tabel 7/6.5.2-21 en figuur 7/6.5.2-38;
- **TIMER0 clock-timing**:
  Tabel 7/6.5.2-22 en figuur 7/6.5.2-39.

In de tabellen 7/6.5.3-1 tot en met -7 wordt een overzicht gegeven van de elektrische eigenschappen van de PIC16C54A-familie, in de tabellen 7/6.5.3-8 tot en met -12 van de PIC16CR57A-familie en in de tabellen 7/6.5.3-13 tot en met -19 van de PIC16CR57A-familie.

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +125°C for automotive, -40°C ≤ T_A ≤ +85°C for industrial and 0°C ≤ T_A ≤ +70°C for commercial Operating voltage V_{DD} = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Supply Voltage	V_{DD}	3.0	-	6.25	V	XT, RC and LP options
		4.5		5.5		HS option
RAM Data Retention Voltage (Note 2)	V_{DR}	1.5	—	—	V	Device in SLEEP mode
V_{DD} start voltage to guarantee Power-On Reset	V_{POR}	—	V_{SS}	—	V	See section on Power-On Reset for details
V_{DD} rise rate to guarantee Power-On Reset	S_{VDD}	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 1)	I_{DD}					XT and RC options (Note 4)
		—	1.7	3.3	mA	F_{OSC} = 4 MHz, V_{DD} = 5.5V
						HS option
		—	2.4	8	mA	F_{OSC} = 10 kHz, V_{DD} = 5.5V
			4.5	16	mA	F_{OSC} = 20 MHz, V_{DD} = 5.5V
Power Down Current (Note 3)	I_{PD}					
WDT enabled		—	4	12	µA	V_{DD} = 3.0V, Commercial
		—	5	14	µA	V_{DD} = 3.0V, Industrial
WDT disabled		—	0.25	4	µA	V_{DD} = 3.0V, Commercial
		—	0.3	5	µA	V_{DD} = 3.0V, Industrial

Tabel 7/6.5.3-4: Gelijkspanningskenmerken van de PIC16C54A-10 en -20 (10, respectievelijk 20 MHz-versie, alleen de voeding).

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ T_A ≤ +125°C for automotive, -40°C ≤ T_A ≤ +85°C for industrial and 0°C ≤ T_A ≤ +70°C for commercial Operating voltage V_{DD} = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Supply Voltage	V_{DD}	2.5	—	6.25	V	XT, RC and LP options
RAM Data Retention Voltage (Note 2)	V_{DR}	1.5	—	—	V	Device in SLEEP mode
V_{DD} start voltage to guarantee Power-On Reset	V_{POR}	—	V_{SS}	—	V	See section on Power-On Reset for details
V_{DD} rise rate to guarantee Power-On Reset	S_{VDD}	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 1)	I_{DD}					XT and RC options (Note 4)
		—	0.5		mA	F_{OSC} = 2 MHz, V_{DD} = 5.5V
						LP option, Commercial
		—	11	27	µA	F_{OSC} = 32 kHz, V_{DD} = 2.5V WDT disabled
						LP option, Industrial
			14	35	µA	F_{OSC} = 32 kHz, V_{DD} = 2.5V WDT disabled
Power Down Current (Note 3)	I_{PD}					
WDT enabled		—	2.5	12	µA	V_{DD} = 2.5V, Commercial
		—	3.5	14	µA	V_{DD} = 2.5V, Industrial
WDT disabled		—	0.25	4	µA	V_{DD} = 2.5V, Commercial
		—	0.3	5	µA	V_{DD} = 2.5V, Industrial

Tabel 7/6.5.3-5: Gelijkspanningskenmerken van de PIC16LC54A-04 (low power 4 MHz-versie, alleen de voeding).

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial Operating voltage VDD = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Input Low Voltage						
I/O ports	VIL	Vss		0.2 VDD	V	Pin at hi-impedance
$\overline{MCLR}$		Vss		0.15 VDD	V	
TOCKI		Vss		0.15 VDD	V	
OSC1		Vss		0.15 VDD	V	RC option only (Note 4)
OSC1		Vss		0.3 VDD	V	XT, HS and LP options
Input High Voltage						
I/O ports	VIH	0.2 VDD+1V		VDD	V	For all VDD (Note 5)
		2.0		VDD	V	4.0V < VDD ≤ 5.5V (Note 5)
$\overline{MCLR}$		0.85 VDD		VDD	V	
TOCKI		0.85 VDD		VDD	V	
OSC1		0.85 VDD		VDD	V	RC option only (Note 4)
OSC1		0.7 VDD		VDD	V	XT, HS and LP options
Input Leakage Current						For VDD ≤ 5.5V
(Note 3)						
I/O ports	IIL	-1	0.5	+1	µA	Vss ≤ VPIN ≤ VDD Pin at hi-impedance
$\overline{MCLR}$		-5			µA	VPIN = Vss +0.25V (Note 2)
$\overline{MCLR}$			0.5	+5	µA	VPIN = VDD (Note 2)
TOCKI		-3	0.5	+3	µA	Vss ≤ VPIN ≤ VDD
OSC1		-3	0.5	+3	µA	Vss ≤ VPIN ≤ VDD XT, HS and LP options
Output Low Voltage						
I/O ports	VOL			0.6	V	IOL = 8.7 mA, VDD = 4.5V
OSC2/CLKOUT (RC option only)				0.6	V	IOL = 1.6 mA, VDD = 4.5V
Output High Voltage						
I/O ports (Note 4)	VOH	VDD-0.7			V	IOH = -5.4 mA, VDD = 4.5V
OSC2/CLKOUT (RC option only)		VDD-0.7			V	IOH = -1.0 mA, VDD = 4.5V

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Total power dissipation as stated under absolute maximum ratings must not be exceeded.

2: The leakage current on the $\overline{MCLR}$/VPP pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: In RC oscillator mode, the OSC1/CLKIN pin is a Schmitt Trigger input. Do not drive the PIC16C54A with an external clock in RC mode.

5: The user may use better of the two specifications.

Tabel 7/6.5.3-6: Gelijkspanningskenmerken van de overige aansluitingen van alle typen PIC16C54A (4, 10 en 20 MHz en 4MHz low-power).

6.5 PIC16/17 typen 8 bit microcontrollers

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	External CLKIN Frequency (Note 1)	DC	—	4	MHz	XT and RC osc mode
			DC	—	4	MHz	HS osc mode (PIC16C5XA-04)
			DC	—	10	MHz	HS osc mode (PIC16C5XA-10)
			DC	—	20	MHz	HS osc mode (PIC16C5XA-20)
			DC	—	200	kHz	LP osc mode
		Oscillator Frequency (Note 1)	DC	—	4	MHz	RC osc mode
			0.1	—	4	MHz	XT osc mode
			4	—	4	MHz	HS osc mode (PIC16C5XA-04)
			4	—	10	MHz	HS osc mode (PIC16C5XA-10)
			4	—	20	MHz	HS osc mode (PIC16C5XA-20)
			5	—	200	kHz	LP osc mode
1	Tosc	External CLKIN Period (Note 1)	250	—	—	ns	XT and RC osc mode
			250	—	—	ns	HS osc mode (PIC16C5XA-04)
			100	—	—	ns	HS osc mode (PIC16C5XA-10)
			50	—	—	ns	HS osc mode (PIC16C5XA-20)
			5.0	—	—	µs	LP osc mode
		Oscillator Period (Note 1)	250	—	—	ns	RC osc mode
			250	—	10,000	ns	XT osc mode
			250	—	250	ns	HS osc mode (PIC16C5XA-04)
			100	—	250	ns	HS osc mode (PIC16C5XA-10)
			50	—	250	ns	HS osc mode (PIC16C5XA-20)
			5	—	200	µs	LP osc mode
2	TCY	Instruction Cycle Time (Note 1)	1.0	—	DC	µs	
3	TosL, TosH	Clock in (OSC1) Low or High Time	50	—	—	ns	XT oscillator
			2.5	—	—	µs	LP oscillator
			10	—	—	ns	HS oscillator
4	TosR, TosF	Clock in (OSC1) Rise or Fall Time	25	—	—	ns	XT oscillator
			50	—	—	ns	LP oscillator
			15	—	—	ns	HS oscillator

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1/CLKIN pin.
When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

Tabel 7/6.5.3-7: Eisen die door de Enhanced PIC16C5xA-typen aan de externe clock worden gesteld (zie ook figuur 7/6.5.2-36).

6.5 PIC16/17 typen 8 bit microcontrollers

Ambient temperature under bias .... -55°C to +125°C
Storage temperature .... - 65°C to +150°C
Voltage on VDD with respect to Vss .... 0 to +7.5V
Voltage on $\overline{MCLR}$ with respect to Vss .... 0 to +14V
Voltage on all other pins with respect to Vss .... -0.6V to (VDD + 0.6V)
Total power dissipation (Note 1) .... 800 mW
Max. current out of Vss pin .... 150 mA
Max. current into VDD pin .... 100 mA
Max. current into an input pin (TOCKI only) .... ±500 µA
Input clamp current, IIK (VI < 0 or VI > VDD) .... ±20 mA
Output clamp current, IOK (V0 < 0 or V0 > VDD) .... ±20 mA
Max. output current sunk by any I/O pin .... 25 mA
Max. output current sourced by any I/O pin .... 20 mA
Max. output current sourced by a single I/O port
PORTA .... 50 mA
PORTB or C .... 100 mA
Max. Output Current sunk by a single I/O port
PORTA .... 50 mA
PORTB or C .... 100 mA

Note 1: Total power dissipation should not exceed 800 mW for the package. Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$$

Tabel 7/6.5.3-8: Maximaal toegelaten waarden voor de PIC16CR57A.

OSC	16CR57A-04	16CR57A-10	16CR57A-20	16LCR57A-04
RC	VDD: 2.5 V to 6.25 V IDD: 3.3 mA Max at 5.5 V IPD: 9 µA max. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 2.5V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.6 µA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 2.5V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.6 µA typ. at 3.0V WDT dis Freq: 4 MHz Max	VDD: 2.5V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.6 µA typ. at 2.5V WDT dis Freq: 4 MHz Max
XT	VDD: 2.5V to 6.25V IDD: 3.3 mA max. at 5.5V IPD: 9 µA max. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 2.5V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.6 µA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 2.5V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.6 µA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 2.5V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.6 µA typ. at 2.5V WDT dis Freq: 4 MHz max.
HS	VDD: 4.5V to 5.5V IDD: 4.8 mA typ. at 5.5V IPD: 0.6 µA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 10 mA max. at 5.5V IPD: 9 µA max. at 3.0V WDT dis Freq: 10 MHz max.	VDD: 4.5V to 5.5V IDD: 20 mA max. at 5.5V IPD: 9 µA max. at 3.0V WDT dis Freq: 20 MHz Max	Do not use in HS mode
LP	VDD: 2.5V to 6.25V IDD: 15 µA typ. at 32 kHz, 3.0V IPD: 0.6 µA typ.at 3.0V WDT dis Freq: 200 kHz max.	Do not use in LP mode	Do not use in LP mode	VDD: 2.5V to 6.25V IDD: 32 µA max. at 32 kHz, 2.5V IPD: 9 µA max. at 2.5V WDT dis Freq: 200 kHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications. It is recommended that the user select the device type that guarantees the specifications required.

Tabel 7/6.5.3-9: Specificaties van diverse oscillator-configuraties en bedrijfsfrequenties voor commerciële typen van de PIC16CR57A.

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS

Standard Operating Conditions (unless otherwise stated)
Operating temperature -40°C ≤ TA ≤ +125°C for automotive,
-40°C ≤ TA ≤ +85°C for industrial and
0°C ≤ TA ≤ +70°C for commercial
Operating voltage: VDD = 4.0V to 6.0V

Characteristic	Sym	Min	Typ†	Max	Units	Conditions
Supply Voltage	VDD	2.5 4.5	-	6.25 5.5	V V	XT, RC and LP osc configuration HS osc configuration
RAM Data Retention Voltage (Note 1)	VDR	1.5	—	—	V	Device in SLEEP mode
VDD start voltage to guarantee Power-On Reset	VPOR	—	Vss	—	V	See section on Power-On Reset for details
VDD rise rate to guarantee Power-On Reset	SVDD	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	IDD					XT and RC options (Note 4)
			1.8	3.3	mA	FOSC = 4 MHz, VDD = 5.5V
						HS option
			4.8	10	mA	FOSC = 10 MHz, VDD = 5.5V
			9.0	20	mA	FOSC = 20 MHz, VDD = 5.5V
						LP osc option, Commercial
			15	32	µA	FOSC = 32 kHz, VDD = 3.0V, WDT disabled
						LP option, Industrial
			19	40	µA	FOSC = 32 kHz, VDD = 3.0V, WDT disabled
Power Down Current (Note 3)	IPD					
WDT enabled		—	4	12	µA	VDD = 3.0V, Commercial
		—	5	14	µA	VDD = 3.0V, Industrial
WDT disabled		—	0.6	9	µA	VDD = 3.0V, Commercial
		—	0.8	12	µA	VDD = 3.0V, Industrial

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as bus loading, oscillator type, bus rate, internal code execution pattern, and temperature also have an impact on the current consumption.

a) The test conditions for all IDD measurements in active operation mode are:
OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD, T0CKI = VDD, $\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.
b) For stand-by current measurements, the conditions are the same, except that the device is in SLEEP mode.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and Vss.

4: RC mode does not include current through Rext. The current through the resistor can be estimated by the formula:
IR = VDD/2Rext (mA) with Rext in kOhm.

Tabel 7/6.5.3-10: Gelijkspanningskenmerken van de PIC16CR57A-04, -10 en -20 (alleen de voeding).

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial Operating voltage: VDD = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Supply Voltage	VDD	2.5	—	6.25	V	XT, RC and LP osc configuration
RAM Data Retention Voltage (Note 1)	VDR	1.5	—	—	V	Device in SLEEP mode
VDD start voltage to guarantee Power-On Reset	VPOR	—	Vss	—	V	See section on Power-On Reset for details
VDD rise rate to guarantee Power-On Reset	SVDD	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	IDD					XT and RC options (Note 4)
			1.8	3.3	mA	FOSC = 4 MHz, VDD = 5.5V
						LP osc option, Commercial
			15	32	µA	FOSC = 32 kHz, VDD = 2.5V, WDT disabled
						LP option, Industrial
			19	40	µA	FOSC = 32 kHz, VDD = 2.5V, WDT disabled
Power Down Current (Note 3)						
WDT enabled	IPD	—	4	12	µA	VDD = 2.5V, Commercial
		—	5	14	µA	VDD = 2.5V, Industrial
WDT disabled		—	0.6	9	µA	VDD = 2.5V, Commercial
		—	0.8	12	µA	VDD = 2.5V, Industrial

Tabel 7/6.5.3-11: Gelijkspanningskenmerken van de PIC16LCR57A-04 (4 MHz, low power-versie, alleen de voeding).

(wordt vervolgd)

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial. Operating voltage VDD = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Input Low Voltage						
I/O ports	VIL	Vss		0.2 VDD	V	Pin at hi-impedance
$\overline{\text{MCLR}}$		Vss		0.15 VDD	V	
TOCKI		Vss		0.15 VDD	V	
OSC1		Vss		0.15 VDD	V	RC option only (Note 4)
OSC1		Vss		0.3 VDD	V	XT, HS and LP options
Input High Voltage						
I/O ports		0.45 VDD		VDD	V	For all VDD (Note 5)
		2.0		VDD	V	4.0V < VDD ≤ 5.5V (Note 5)
		0.36 VDD		VDD	V	VDD > 5.5V
$\overline{\text{MCLR}}$		0.85VDD		VDD	V	
TOCKI		0.85 VDD		VDD	V	RC option only (Note 4)
OSC1		0.85 VDD		VDD	V	XT, HS and LP options
OSC1		0.7 VDD		VDD	V	
Input Leakage Current (Note 3)						**For VDD ≤ 5.5V**
I/O ports	IIL	-1	0.5	+1	µA	VSS ≤ VPIN ≤ VDD Pin at hi-impedance
$\overline{\text{MCLR}}$		-5			µA	VPIN = VSS +0.25V (Note 2)
$\overline{\text{MCLR}}$			0.5	+5	µA	VPIN = VDD (Note 2)
TOCKI		-3	0.5	+3	µA	VSS ≤ VPIN ≤ VDD
OSC1		-3	0.5	+3	µA	VSS ≤ VPIN ≤ VDD XT, HS and LP options
Output Low Voltage						
I/O ports	VOL			0.6	V	IOL = 8.7 mA, VDD = 4.5V
OSC2/CLKOUT (RC option only)				0.6	V	IOL = 1.6 mA, VDD = 4.5V
Output High Voltage						
I/O ports (Note 4)	VOH	VDD-0.7			V	IOH = -5.4 mA, VDD = 4.5V
OSC2/CLKOUT (RC option only)		VDD-0.7			V	IOH = -1.0 mA, VDD = 4.5V

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Total power dissipation as stated under absolute maximum ratings must not be exceeded.

2: The leakage current on the $\overline{\text{MCLR}}$/VPP pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: In RC oscillator mode, the OSC1/CLKIN pin is a Schmitt Trigger input. Do not drive the PIC16C54A with an external clock in RC mode.

5: The user may use better of the two specifications.

Tabel 7/6.5.3-12: Gelijkspanningskenmerken van de overige aansluitingen van alle typen PIC16CR57A (4, 10 en 20 MHz en low-power 4 MHz).

6.5 PIC16/17 typen 8 bit microcontrollers

Ambient temperature Under Bias	- 55°C to +125°C
Storage temperature	- 65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	0 to +7.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS}	0 to +14V
Voltage on all other pins with respect to V_{SS}	-0.6V to (V_{DD} + 0.6V)
Total power dissipation (Note 1)	800 mW
Max. current out of V_{SS} pin	150 mA
Max. current into V_{DD} pin	100 mA
Max. current into an input pin (T0CKI only)	±500 μA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD})	±20 mA
Output clamp current, I_{OK} (V_0 < 0 or V_0 > V_{DD})	±20 mA
Max. output current sunk by any I/O pin	25 mA
Max. output current sourced by any I/O pin	20 mA
Max. output current sourced by a single I/O port	
PORTA	50 mA
PORTB	100 mA
Max. Output Current sunk by a single I/O port	
PORTA	50 mA
PORTB	100 mA

Note 1: Total power dissipation should not exceed 800 mW for the package. Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$$

Tabel 7/6.5.3-13: Maximaal toegelaten waarden voor de PIC16C58A en PIC16CR58A.

OSC	16C58A-04	16C58A-10	16C58A-20	16LC58A-04
RC	V_{DD}: 3.0V to 6.25V I_{DD}: 2.5 mA max. at 5.5V I_{PD}: 4 μA max. at 3.0V WDT dis Freq: 4 MHz max.	V_{DD}: 3.0V to 6.25V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	V_{DD}: 3.0V to 6.25V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz Max	V_{DD}: 2.5V to 6.25V I_{DD}: 0.5 mA typ. at 5.5V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 2 MHz max.
XT	V_{DD}: 3.0V to 6.25V I_{DD}: 2.5 mA max. at 5.5V I_{PD}: 4 μA max. at 3.0V WDT dis Freq: 4 MHz max.	V_{DD}: 3.0V to 6.25V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz Max	V_{DD}: 3.0V to 6.25V I_{DD}: 1.8 mA typ. at 5.5V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz Max	V_{DD}: 2.5V to 6.25V I_{DD}: 0.5 mA typ. at 5.5V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 2 MHz max.
HS	V_{DD}: 4.5V to 5.5V I_{DD}: 1.9 mA typ. at 5.5V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 8 mA max. at 5.5V I_{PD}: 4 μA max. at 3.0V WDT dis Freq: 10 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 17 mA max. at 5.5V I_{PD}: 4 μA max. at 3.0V WDT dis Freq: 20 MHz max.	Do not use in HS mode
LP	V_{DD}: 3.0V to 6.25V I_{DD}: 15 μA typ. at 32 kHz, 3.0V I_{PD}: 0.25 μA typ. at 3.0V WDT dis Freq: 200 kHz max.	Do not use in LP mode	Do not use in LP mode	V_{DD}: 2.5V to 6.25V I_{DD}: 28 μA max. at 32 kHz, 2.5V I_{PD}: 4 μA max. at 2.5V WDT dis Freq: 200 kHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications. It is recommended that the user select the device type that guarantees the specifications required.

Tabel 7/6.5.3-14: Overzicht van de specificaties voor diverse oscillator-configuraties en bedrijfsfrequenties voor commerciële typen van de PIC16C58A.

6.5 PIC16/17 typen 8 bit microcontrollers

OSC	16CR58A-04	16CR58A-10	16CR58A-20	16LCR58A-04
RC	VDD: 3.0V to 6.25V IDD: 2.5 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 3.0V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 3.0V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz Max	VDD: 2.5V to 6.25V IDD: 0.5 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 2 MHz max.
XT	VDD: 3.0V to 6.25V IDD: 2.5 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 3.0V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz Max	VDD: 3.0V to 6.25V IDD: 1.8 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz Max	VDD: 2.5V to 6.25V IDD: 0.5 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 2 MHz max.
HS	VDD: 4.5V to 5.5V IDD: 1.9 mA typ. at 5.5V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 8 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 10 MHz max.	VDD: 4.5V to 5.5V IDD: 17 mA max. at 5.5V IPD: 4 μA max. at 3.0V WDT dis Freq: 20 MHz max.	Do not use in HS mode
LP	VDD: 3.0V to 6.25V IDD: 15 μA typ. at 32 kHz, 3.0V IPD: 0.25 μA typ. at 3.0V WDT dis Freq: 200 kHz max.	Do not use in LP mode	.Do not use in LP mode	VDD: 2.5V to 6.25V IDD: 28 μA max. at 32 kHz, 2.5V IPD: 4 μA max. at 2.5V WDT dis Freq: 200 kHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications. It is recommended that the user select the device type that guarantees the specifications required.

Tabel 7/6.5.3-15: Overzicht van de specificaties voor diverse oscillator-configuraties en bedrijfsfrequenties voor commerciële typen van de PIC16CR58A.

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial. Operating voltage: VDD = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Supply Voltage	VDD	3.0	—	6.25	V	XT, RC and LP osc configuration
RAM Data Retention Voltage (Note 1)	VDR	1.5	—	—	V	Device in SLEEP mode
VDD start voltage to guarantee Power-On Reset	VPOR	—	Vss	—	V	See section on Power-On Reset for details
VDD rise rate to guarantee Power-On Reset	SVDD	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	IDD					XT and RC options (Note 4)
			1.9	2.5	mA	Fosc = 4 MHz, VDD = 5.5V
						LP osc option, Commercial
			15		μA	Fosc = 32 kHz, VDD = 3.0V, WDT disabled
						LP option, Industrial
			18		μA	Fosc = 32 kHz, VDD = 3.0V, WDT disabled
Power Down Current (Note 3)	IPD					
WDT enabled		—	4	12	μA	VDD = 3.0V, Commercial
		—	5	14	μA	VDD = 3.0V, Industrial
WDT disabled		—	0.25	4	μA	VDD = 3.0V, Commercial
		—	0.3	5	μA	VDD = 3.0V, Industrial

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as bus loading, oscillator type, bus rate, internal code execution pattern, and temperature also have an impact on the current consumption.

a) The test conditions for all IDD measurements in active operation mode are:
OSC1=external square wave, from rail to rail; all I/O pins tristated, pulled to VDD, T0CKI = VDD, $\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.
b) For stand-by current measurements, the conditions are the same, except that the device is in SLEEP mode.

3: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and Vss.

4: RC mode does not include current through Rext. The current through the resistor can be estimated by the formula:
IR = VDD/2Rext (mA) with Rext in kOhm.

Tabel 7/6.5.3-16: Gelijkspanningskenmerken van de PIC16C58A/CR58A-04 (4 MHz commerciële versie, alleen de voeding).

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial Operating voltage: VDD = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Supply Voltage	VDD	3.0 4.5	—	6.25 5.5	V	XT, RC and LP osc configuration HS osc configuration
RAM Data Retention Voltage (Note 1)	VDR	1.5	—	—	V	Device in SLEEP mode
VDD start voltage to guarantee Power-On Reset	VPOR	—	Vss	—	V	See section on Power-On Reset for details
VDD rise rate to guarantee Power-On Reset	SVDD	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	IDD					XT and RC options (Note 4)
			1.8		mA	FOSC = 4 MHz, VDD = 5.5V
						HS option
			2.5	8	mA	FOSC = 10 MHz, VDD = 5.5V
			4.7	17	mA	FOSC = 20 MHz, VDD = 5.5V
Power Down Current (Note 3)	IPD					
WDT enabled		—	4	12	µA	VDD = 3.0V, Commercial
		—	5	14	µA	VDD = 3.0V, Industrial
WDT disabled		—	0.25	4	µA	VDD = 3.0V, Commercial
		—	0.3	5	µA	VDD = 3.0V, Industrial

Tabel 7/6.5.3-17: Gelijkspanningskenmerken van de PIC16C58A/CR58A-10 en -20 (10, respectievelijk 20 MHz-versie, alleen de voeding).

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS
POWER SUPPLY PINS

Standard Operating Conditions (unless otherwise stated)
Operating temperature -40°C ≤ T_A ≤ +125°C for automotive,
-40°C ≤ T_A ≤ +85°C for industrial and
0°C ≤ T_A ≤ +70°C for commercial
Operating voltage: V_{DD} = 4.0V to 6.0V

Characteristic	Sym	Min	Typ†	Max	Units	Conditions
Supply Voltage	V_{DD}	2.5	-	6.25	V	XT, RC and LP osc configuration
RAM Data Retention Voltage (Note 1)	V_{DR}	1.5	—	—	V	Device in SLEEP mode
V_{DD} start voltage to guarantee Power-On Reset	V_{POR}	—	Vss	—	V	See section on Power-On Reset for details
V_{DD} rise rate to guarantee Power-On Reset	S_{VDD}	0.05*	—	—	V/ms	See section on Power-On Reset for details
Supply Current (Note 2)	I_{DD}					XT and RC options (Note 4)
			0.5		mA	F_{OSC} = 2 MHz, V_{DD} = 5.5V
						LP option, Commercial
			12	28	µA	F_{OSC} = 32 kHz, V_{DD} = 2.5V, WDT disabled
						LP option, Industrial
			15	37	µA	F_{OSC} = 32 kHz, V_{DD} = 2.5V, WDT disabled
Power Down Current (Note 3)	I_{PD}					
WDT enabled		—	2.5	12	µA	V_{DD} = 2.5V, Commercial
		—	3.5	14	µA	V_{DD} = 2.5V, Industrial
WDT disabled		—	0.25	4	µA	V_{DD} = 2.5V, Commercial
		—	0.3	5	µA	V_{DD} = 2.5V, Industrial

Tabel 7/6.5.3-18: Gelijkspanningskenmerken van de PIC16LC58A/LCR58A-04 (low power 4 MHz-versie, alleen de voeding).

6.5 PIC16/17 typen 8 bit microcontrollers

DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for automotive, -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial Operating voltage VDD = 4.0V to 6.0V					
Characteristic	**Sym**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
Input Low Voltage						
I/O ports	VIL	Vss		0.2 VDD	V	Pin at hi-impedance
$\overline{MCLR}$		Vss		0.15 VDD	V	
TOCKI		Vss		0.15 VDD	V	
OSC1		Vss		0.15 VDD	V	RC option only (Note 4)
OSC1		Vss		0.3 VDD	V	XT, HS and LP options
Input High Voltage						
I/O ports		0.45 VDD		VDD	V	For all VDD (Note 5)
		2.0		VDD	V	4.0V < VDD ≤ 5.5V (Note 5)
		0.36 VDD		VDD	V	VDD > 5.5V
$\overline{MCLR}$		0.85 VDD		VDD	V	
TOCKI		0.85 VDD		VDD	V	RC option only (Note 4)
OSC1		0.85 VDD		VDD	V	XT, HS and LP options
OSC1		0.7 VDD		VDD	V	
Input Leakage Current (Note 3)						**For VDD ≤ 5.5V**
I/O ports	IIL	-1	0.5	+1	µA	Vss ≤ VPIN ≤ VDD Pin at hi-impedance
$\overline{MCLR}$		-5			µA	VPIN = Vss +0.25V (Note 2)
$\overline{MCLR}$			0.5	+5	µA	VPIN = VDD (Note 2)
TOCKI		-3	0.5	+3	µA	Vss ≤ VPIN ≤ VDD
OSC1		-3	0.5	+3	µA	Vss ≤ VPIN ≤ VDD XT, HS and LP options
Output Low Voltage						
I/O ports	VOL			0.6	V	IOL = 8.7 mA, VDD = 4.5V
OSC2/CLKOUT (RC option only)				0.6	V	IOL = 1.6 mA, VDD = 4.5V
Output High Voltage						
I/O ports (Note 4)	VOH	VDD-0.7			V	IOH = -5.4 mA, VDD = 4.5V
OSC2/CLKOUT (RC option only)		VDD-0.7			V	IOH = -1.0 mA, VDD = 4.5V

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Total power dissipation as stated under absolute maximum ratings must not be exceeded.

2: The leakage current on the $\overline{MCLR}$/VPP pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

4: In RC oscillator mode, the OSC1/CLKIN pin is a Schmitt Trigger input. Do not drive the PIC16C54A with an external clock in RC mode.

5: The user may use better of the two specifications.

Tabel 7/6.5.3-19: Gelijkspanningskenmerken van de overige aansluitingen van alle typen PIC16C58A/CR58A (4, 10 en 20 MHz en 4 MHz, low-power).

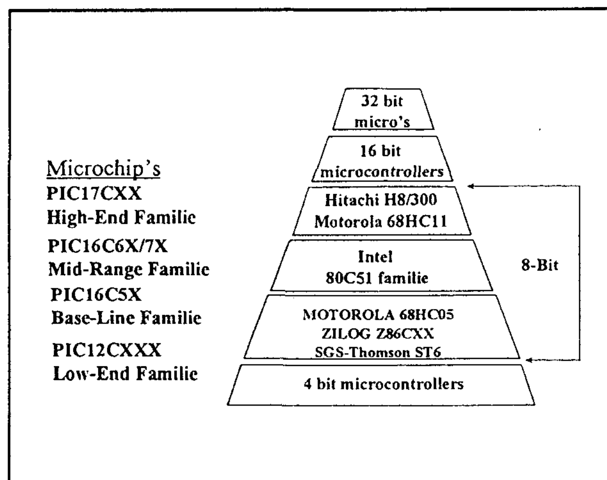
6.5 PIC16/17 typen 8 bit microcontrollers

7/6.5.4

Achtergrond-informatie van de PIC 12-familie

Inleiding

De 8 bit PIC microcontroller-families van de firma Microchip zijn ondertussen zeer goede bekenden van de ontwerpers van elektronische schakelingen geworden. In figuur 7/6.5.4-1 is de positionering van de PIC-microcontrollers ten opzichte van andere typen te zien. Afhankelijk van de gewenste aansluitmogelijkheden worden de PIC's geleverd in behuizingen met 8, 14, 18/20, 28, 40/44 of 64/68 pennen. Alle families hebben als overeenkomstige kenmerken:



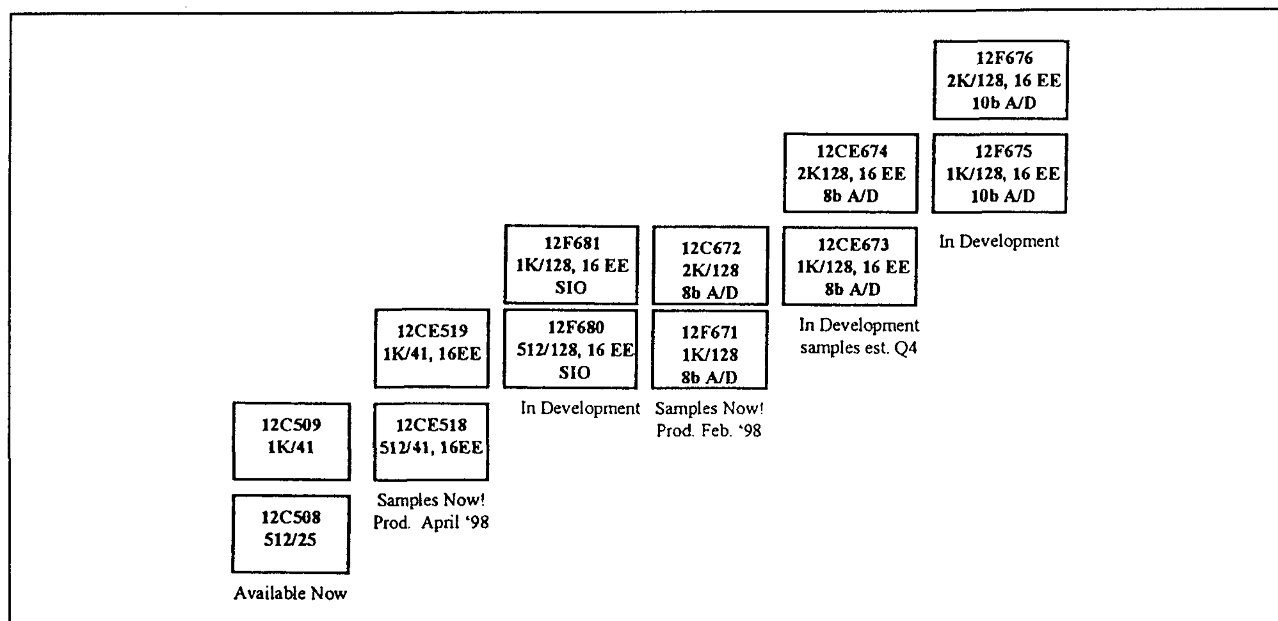
Figuur 7/6.5.4-1: De positie van de 8 bit PIC-microcontrollers ten opzichte van andere typen en merken.

- goedkope 8 bit RISC microcontrollers
- hoge snelheid: 5 MIPS
- geringe dissipatie: 40 µA bij 32 kHz
- grote uitgangsstroom
- programma-geheugen: 384 tot 16 k woorden
- data-geheugen op SRAM gebaseerd: 25 tot 902 bytes
- compatibel in dezelfde familie
- maximaal 35 instructies (gemakkelijk te leren)
- eenvoudig te hanteren ontwikkelgereedschappen

In dit gedeelte worden de 8-pens microcontrollers van de PIC12-serie behandeld. Deze kleinste familie is nog goedkoper dan de andere typen van Microchip en heeft op de print slechts 39 mm^2 nodig. Als gevolg van de 8-pens behuizing zijn echter slechts 6 pennen beschikbaar als in- of uitgang, waarbij dan de interne clock-oscillator moet worden gebruikt.

In figuur 7/6.5.4-2 zijn de plannen van de complete PIC12xxx-serie te zien. Bij de PIC12Cxx-typen wordt EPROM gebruikt voor de opslag van het programma en SRAM voor de data. Deze typen worden met en zonder kwartsvenstertje geleverd, waardoor de eerste wél en de tweede niet UV-wisbaar is (deze is dus OTP: One-Time Programmeerbaar). Bij de PIC12CExxx-typen bevindt zich naast de EPROM en de SRAM nog een EEPROM die dus elektrisch kan worden ge(her)programmeerd. Bij de PIC12Fxxx-typen is de EPROM vervangen door een flash-geheugen waarvan de inhoud elektrisch kan worden gewijzigd, zodat een venstertje niet langer nodig is. Een groot bijkomend voordeel is dat het programmeren van de schakeling dan op de print zelf kan plaatsvinden.

6.5 PIC-typen 8 bit microcontrollers

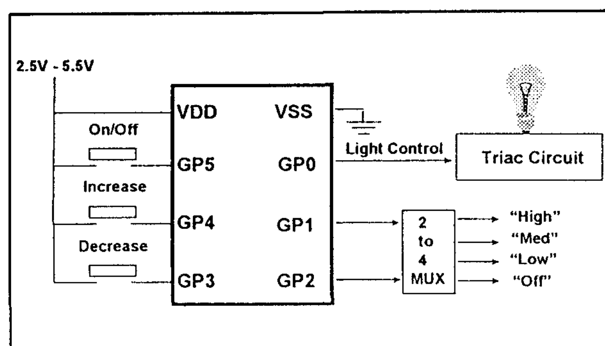


Figuur 7/6.5.4-2: Overzicht van de ontwikkelingen in de 8-pens PIC12xxx-familie.

Op dit moment zijn de PIC12C5xx- en PIC12CE5xx-families en de PIC12C67x-familie leverbaar.

Toelichting

De PIC12C509 heeft 1 kB EPROM plus 41 bytes SRAM aan boord; de IC12CE518 is voorzien van 512 bytes EPROM, 41 bytes SRAM en 16 bytes EEPROM. Verder zijn er typen met een 8 of 10 bit A/D-omzetter of seriële I/O.



Figuur 7/6.5.4-3: Toepassingsvoorbeeld van een PIC12C5xx als stroomregelaar.

7/6.5.5

Type-beschrijving van de PIC12C5xx-typen

Algemene gegevens

Inleiding

De PIC12C5xx is, met slechts 8 aansluitpennen, de kleinste familie van Microchip's 8 bit, volledig statische, op EPROM/ROM gebaseerde CMOS microcontrollers. Deze familie is gebaseerd op de Enhanced PIC16C5x-familie en pin- en software-compatibel met de PIC12CE5xx-familie die over een extra EEPROM beschikt. Er wordt gebruik gemaakt van een RISC architectuur met slechts 33 single word/single cycle instructies. Alle instructies worden uitgevoerd in één cyclus (1 µs), behalve branch-instructies, waar twee cycli voor nodig zijn. De 12 bit brede instructies zijn in sterk symmetrisch, waardoor een 2:1 code compressie ten opzichte van andere 8 bit microcontrollers ontstaat. De PIC12C5xx-produkten zijn speciaal ingericht om de systeemkosten en de eisen die aan de voeding worden gesteld te beperken. De Power-On Reset (POR) en de Device Reset Timer (DRT) maken externe reset-schakelingen overbodig. Er kan uit vier oscillator-configuraties worden gekozen, inclusief de INTRC interne oscillator mode en de stroombesparende LP (Low Power) oscillator. Verder kan gebruik worden gemaakt van de SLEEP mode, de Watchdog Timer en code beveiligingen.

De UV-wisbare CERDIP versies zijn ideaal voor het ontwikkelen van code, terwijl de goedkopere OTP-versies bruikbaar zijn voor de produktie (bij alle aantallen).

De PIC12C5xx-produkten worden ondersteund door een volledig toegeruste macro-assembler, een software simulator, een in-circuit emulator, een C-compiler, fuzzy-logic tools, een goedkope ontwikkel-programmer en een complete programmer. Alle tools zijn bruikbaar op IBM PC's en vergelijkbare typen.

Algemene gegevens

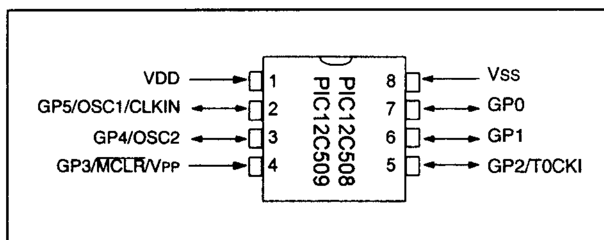
- behandelde typen: PIC12C508 en PIC12C509
- 33 single-word instructies
- één cyclus per instructie (1 µs), behalve branches (2 cycli)
- snelheid: DC - 4 MHz clock input
- DC - 1 µs instructie-cyclus
- instructies: 12 bit breed
- data-pad: 8 bit breed
- 7 special function hardware registers
- 2-niveaus diepe hardware stack
- directe, indirecte en relatieve adresseringsmodes voor data en instructies
- interne 4 MHz RC-oscillator met programmeerbare calibratie
- in-circuit seriële programmering
- 8 bit real-time clock/counter (TMR0) met 8 bit programmeerbare prescaler
- power-on reset (POR) en device reset timer (DRT)
- watchdog timer (WDT) met eigen on-chip RC-oscillator
- programmeerbare code-beveiliging
- energiebesparende SLEEP-mode
- ontwaken uit SLEEP-mode bij signaalverandering op pen
- interne "zwakke" optrekking van I/O-pennen
- interne pull-up van $\overline{MCLR}$-pen

6.5 PIC-typen 8 bit microcontrollers

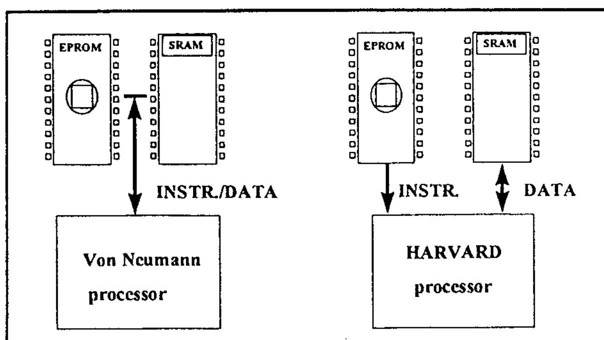
	Clock	Memory		Peripherals	Features							
	Maximum Frequency of Operation (MHz)	Program Memory EPROM	Data Memory (bytes)	Timer Module(s)	Wake-up from SLEEP on pin change	I/O Pins	Input Pins	Internal Pull-ups	Voltage Range (Volts)	In-Circuit Serial Programming	Number of Instructions	Packages
PIC12C508	4	512	25	TMR0	Yes	5	1	Yes	2.5-5.5	Yes	33	8-pin PDIP, 8-pin SOIC
PIC12C509	4	1024	41	TMR0	Yes	5	1	Yes	2.5-5.5	Yes	33	8-pin PDIP, 8-pin SOIC

All PIC12C5XX devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.
All PIC12C5XX devices use serial programming with data pin GP0 and clock pin GP1.

Figuur 7/6.5.5-1: Kenmerken van de PIC12C5xx-typen.



Figuur 7/6.5.5-2: Aansluitingen van de PIC12C508 en PIC12C509 (8-pens PDIP, SOIC en CER-DIP uitvoering (met venster).



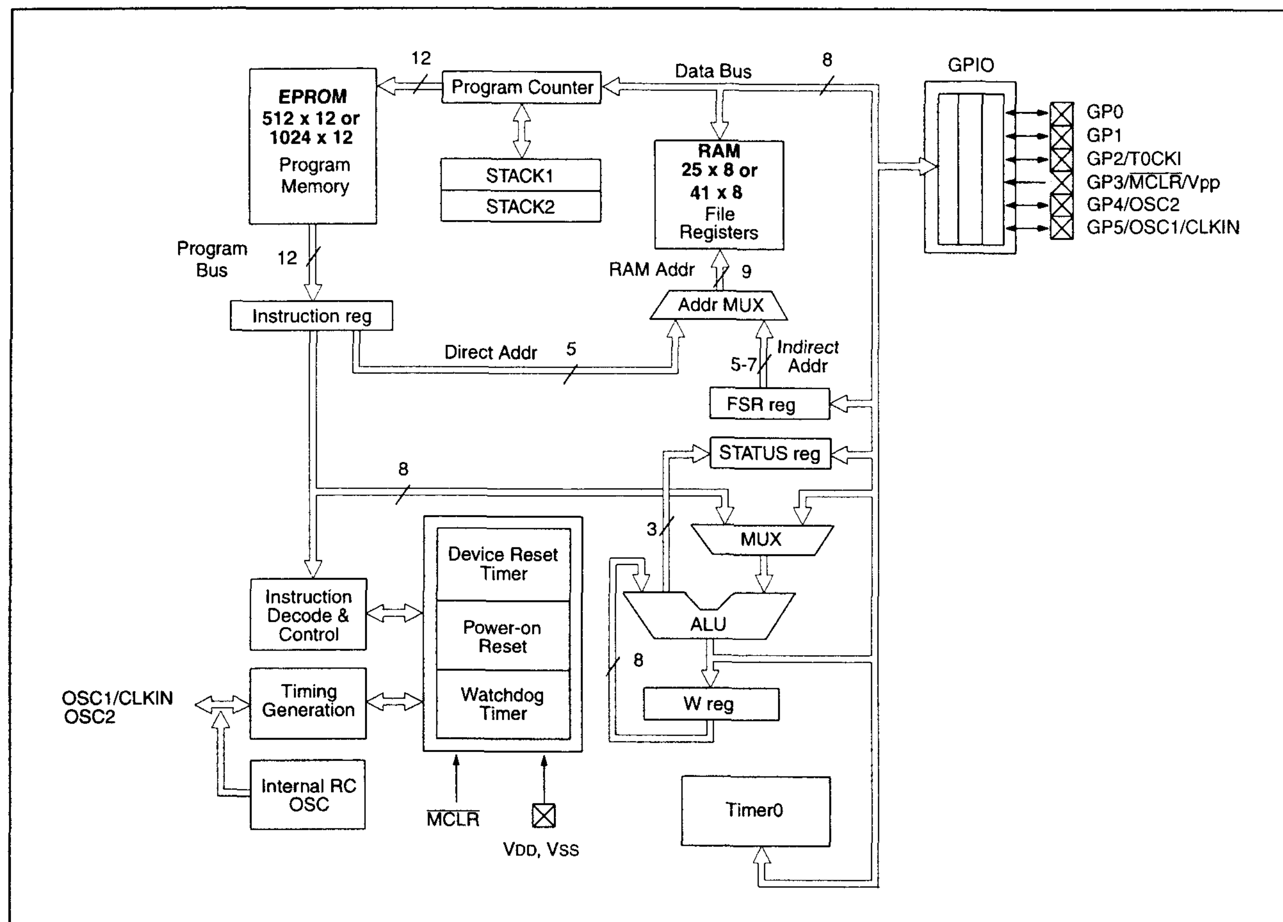
Figuur 7/6.5.5-3: Verschil tussen de gebruikelijke Von Neumann architectuur en de bij de PIC's gebruikte Harvard architectuur.

- selecteerbare oscillator-opties:
  - INTRC: interne 4 MHz RC-oscillator
  - EXTRC: externe RC-oscillator
  - XT: standaard kristal/resonator
  - LP: energie besparend laagfrequent kristal
- CMOS: volledig statisch ontwerp
- voedingsspanning:
  commercieel en industrieel:
  2,5 V - 5,5 V
  uitgebreid: 3,0 V - 5,5 V
- laag energie-verbruik:
  - <2 mA typ. (5 V, 4 MHz)
  - 15 μA typ. (3 V, 32 kHz)
  - <1 μA typ. (standby)
- fabrikant: Microchip Technology Inc.

Toepassingen

De PIC12C5xx-serie kan bijvoorbeeld uitstekend worden toegepast in toepassingen voor persoonlijke verzorging, beveiligingssystemen en draagbare zenders/ontvangers. Door de EPROM-technologie is aanpassing van de programma's (zender-codes, instellingen, ontvangstfrequenties, enzovoorts) zeer snel en gemakkelijk realiseerbaar. De kleine afmetingen (al dan niet voor oppervlaktemontage) maken deze microcontrollers perfect voor toepassingen met beperkte ruimte. De lage prijs, geringe dissipatie, hoog prestatieniveau, groot gebruiksgemak en flexibele I/O zorgen voor een grote aantrekkingskracht.

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-4: Vereenvoudigd blokschema van de PIC12C5xx microcontrollers.

Architectuur

Inleiding

Prestaties als van de PIC12C5xx-familie komen normaal alleen voor bij RISC microprocessoren. Om te beginnen gebruikt de PIC12C5xx een Harvard architectuur, waarbij de toegang tot programma en data via aparte bussen plaatsvindt. Hierdoor wordt de bandbreedte verbeterd ten opzichte van de traditionele von Neumann architectuur (waarbij programma en data via dezelfde bus worden opgehaald). Door programma- en data-geheugen van elkaar te scheiden kunnen instructies ook andere afmetingen hebben dan het 8 bit brede datawoord. Met een 12 bit brede programma-bus kan een 12 bit instructie in één enkele cyclus worden opgehaald. In figuur 7/6.5.5-3 is te zien wat hiermee wordt bedoeld. Met een tweetraps pijplijn overlappen fetch en executie van instructies elkaar. Als gevolg daarvan worden alle instructies in één enkele cyclus (1 µs bij 4 MHz) uitgevoerd, met uitzondering van program-branches.

De PIC12C508 adresseert 512 x 12 programma-geheugen en de PIC12C509 adresseert 1 k x 12. Het gehele programmageheugen is intern. De PIC12C5xx kan zijn register-files en data-geheugen direct of indirect adresseren. Alle registers voor speciale functies (waaronder ook de programcounter) zijn in het data-geheugen opgenomen. Door de zeer orthogonale (symmetrische) instructieset van de PIC12C5xx is het mogelijk om elke willekeurige operatie, met

wat voor adresseringsmode dan ook, uit te voeren op alle registers. Een PIC12C5xx-microcontroller bevat een 8 bit ALU en werkregister. De ALU is een algemene rekenkundige eenheid waarmee rekenkundige en Boole'se bewerkingen tussen data in het werkregister en data in een willekeurige register-file kunnen worden uitgevoerd. De ALU is 8 bit breed en kan optellen, aftrekken, schuiven en logische operaties uitvoeren. Tenzij anders vermeld zijn rekenkundige operaties two's complement van aard. Bij 2-operand instructies is gewoonlijk één operand het werkregister (W). De andere is óf een file-register óf een immediate constante. Bij 1-operand instructies is de operand het W-register of een file-register. Het W-register is een 8 bit werkregister dat voor ALU-operaties wordt gebruikt. Het is niet adresseerbaar. Afhankelijk van de uitgevoerde instructie kan de ALU de waarden van de Carry (C), Digit Carry (DC) en Zero (Z) bits in het statusregister beïnvloeden. De C en DC bits werken bij aftrekken respectievelijk als borrow en digit borrow out-bit, zie bijvoorbeeld de SUBWF en ADDWF instructies.

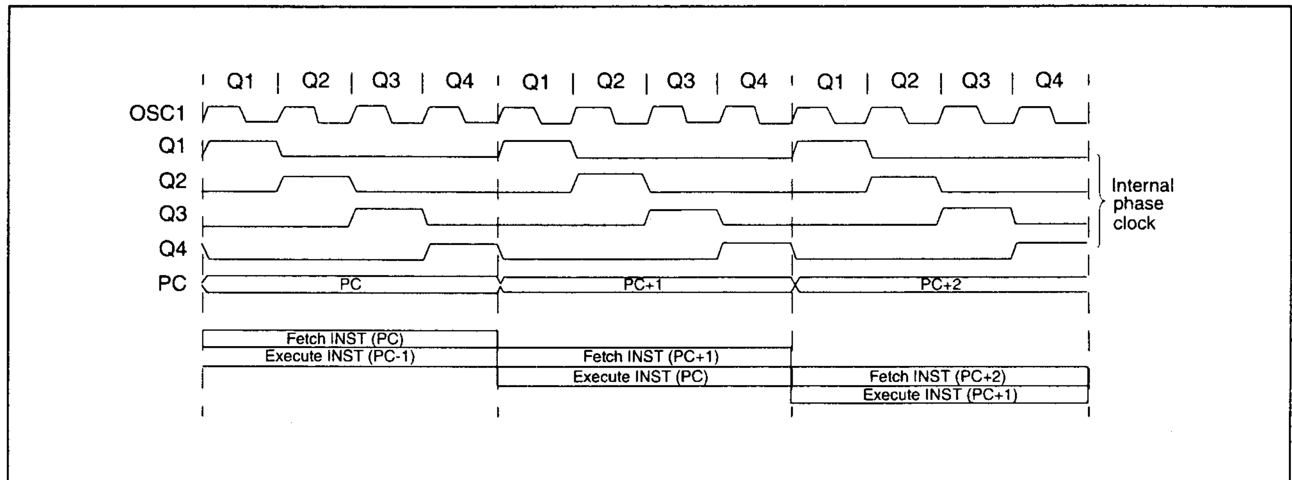
Clock-schema/Instructie-cyclus
Het clock-ingangssignaal (OSC1/CLKIN) wordt intern door vier gedeeld om vier niet-overlappende kwadratuur clock-signalen te genereren (Q1, Q2, Q3 en Q4). Intern wordt de program-counter bij elke Q1 met één verhoogd, terwijl de instructie uit het programma-geheugen wordt opgehaald om op Q4 in het instructie-register te worden opgeborgen.
De instructie wordt gedurende de volgende Q1 tot en met Q4 gedecodeerd en uitgevoerd. In de figuren 7/6.5.5-5 en 7/6.5.5-6 is het verband tussen de clocksignalen en de instructies te zien.

Name	DIP Pin #	SOIC Pin #	I/O/P Type	Buffer Type	Description
GP0	7	7	I/O	TTL/ST	Bi-directional I/O port/ serial programming data. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. This buffer is a Schmitt Trigger input when used in serial programming mode.
GP1	6	6	I/O	TTL/ST	Bi-directional I/O port/ serial programming clock. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. This buffer is a Schmitt Trigger input when used in serial programming mode.
GP2/T0CKI	5	5	I/O	ST	Bi-directional I/O port. Can be configured as T0CKI.
GP3/$\overline{MCLR}$/VPP	4	4	I	TTL	Input port/master clear (reset) input/programming voltage input. When configured as $\overline{MCLR}$, this pin is an active low reset to the device. Voltage on $\overline{MCLR}$/VPP must not exceed VDD during normal device operation. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. Weak pull-up always on if configured as $\overline{MCLR}$
GP4/OSC2	3	3	I/O	TTL	Bi-directional I/O port/oscillator crystal output. Connections to crystal or resonator in crystal oscillator mode (XT and LP modes only, GPIO in other modes).
GP5/OSC1/CLKIN	2	2	I/O	TTL/ST	Bidirectional IO port/oscillator crystal input/external clock source input (GPIO in Internal RC mode only, OSC1 in all other oscillator modes). TTL input when GPIO, ST input in external RC oscillator mode.
VDD	1	1	P	—	Positive supply for logic and I/O pins
VSS	8	8	P	—	Ground reference for logic and I/O pins

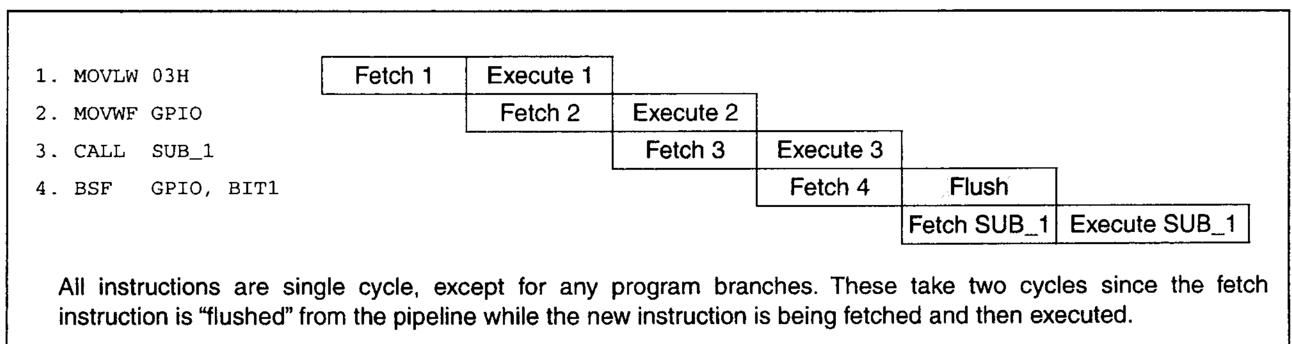
Legend: I = input, O = output, I/O = input/output, P = power, — = not used, TTL = TTL input, ST = Schmitt Trigger input

Tabel 7/6.5.5-1: **Aansluitingen en signaalfuncties van de PIC12C508 en PIC12C509.**

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-5: Clock/instructie-cyclus.



Figuur 7/6.5.5-6: Voorbeeld van de instructiestroom in de pijplijn.

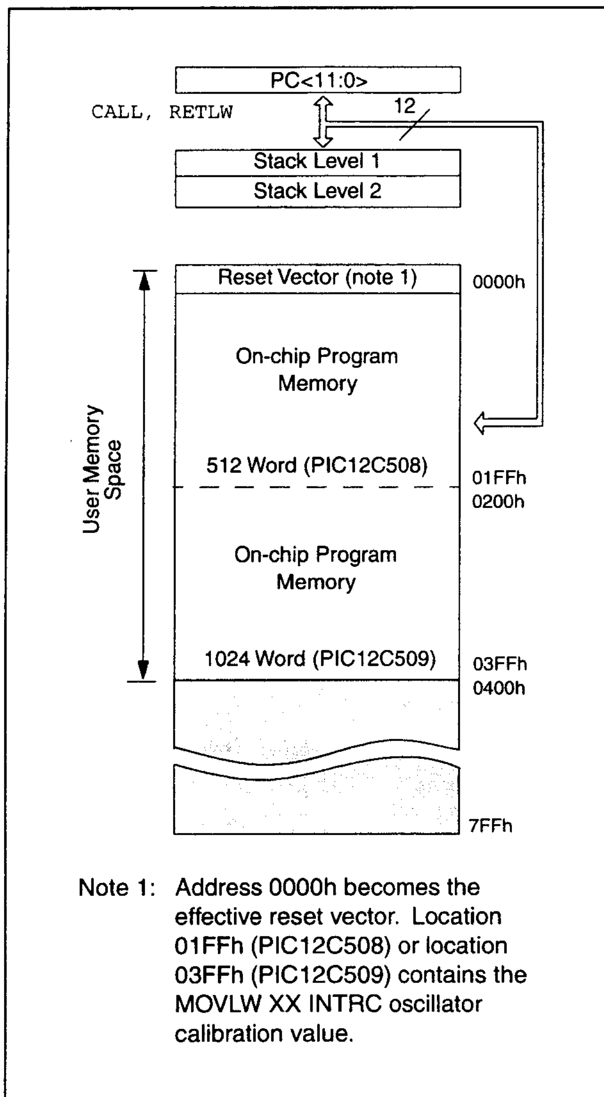
Instructie-afhandeling/pijplijnen
Een instructie-cyclus bestaat uit vier Q-cycli (Q1, Q2, Q3 en Q4). Het ophalen en uitvoeren van de instructie worden gepijplijnd. Voor het ophalen is dus één instructie-cyclus nodig, terwijl het decoderen en uitvoeren nog een instructie-cyclus duurt. Door het pijplijnen is echter voor elke instructie slechts één cyclus nodig. Als door een instructie de program-counter verandert (bijvoorbeeld door GOTO) zijn twee cycli nodig om de instructie uit te voeren (zie figuur 7/6.5.5-6). Een ophaal-cyclus (fetch) begint met het ophogen van de program-counter (PC) bij Q1. In de executie-cyclus wordt de opgehaalde instructie bij Q1 in het Instructie Register (IR) gelatched. Deze instructie wordt vervolgens gedurende Q2, Q3 en Q4 gedecodeerd en uitgevoerd.

Het data-geheugen wordt bij Q2 gelezen (operand read) en bij Q4 weggeschreven (destination write).

Geheugen-organisatie

Inleiding
Het geheugen van de PIC12C5xx is georganiseerd in programma-geheugen en data-geheugen. Wanneer meer dan 512 bytes programma-geheugen aanwezig zijn, wordt "paging" toegepast. Toegang tot de programma-geheugen-pagina's wordt verkregen met behulp van een bit in het STATUS-register. Dit is bijvoorbeeld het geval voor de PIC12C509 die meer dan 32 registers bevat. Data-geheugenbanken worden bereikt met behulp van het File Select Register (FSR).

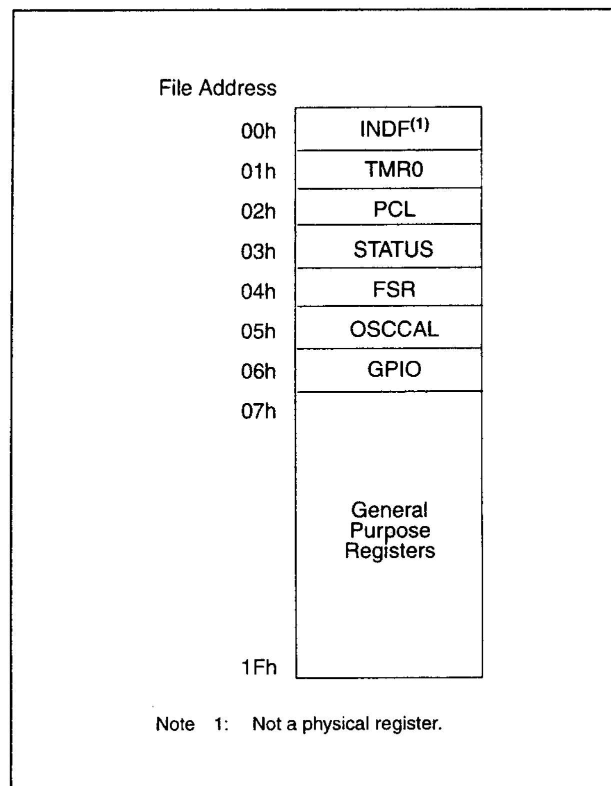
6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-7: Organisatie van het programma-geheugen en de stack voor de PIC12C5xx.

Programma-geheugen

De PIC12C508 en PIC12C509 hebben allebei een 12 bit Program Counter, waarmee het mogelijk is 2 k geheugenwoorden van elk 12 bit te adresseren.
Bij de PIC12C508 zijn alleen de eerste 512 x 12 (0000h - 01FFh) fysiek geïmplementeerd en bij de PIC12C509 alleen de eerste 1 k x 12 (0000h - 03FFh) (zie figuur 7/6.5.5-7). Het adresseren van een lokatie boven deze grenzen zal herhaald betreden van dezelfde ruimte tot gevolg hebben (512 x 12 bij de PIC12C508 en 1 k x 12 bij de PIC12C509). De effectieve resetvector ligt bij 0000h. Lokatie 01FFh (PIC12C508) of 03FFh (PIC12C508) bevat de calibratiewaarde van de interne clock-oscillator. Deze waarde moet nooit worden overschreven.



Figuur 7/6.5.5-8: Register-file van de PIC12C508.

Data-geheugen

Het data-geheugen bestaat uit registers (of bytes) RAM. Daarom wordt het datageheugen gespecificeerd door een registerfile. De register-file is verdeeld in twee functionele groepen: speciale functie-registers en algemene (general purpose) registers.
Tot de special function registers behoren het TMR0 Register, de Program Counter (PC), het Status Register, de I/O Registers (poorten) en het File Select Register (FSR). Bovendien worden er speciale registers ge-

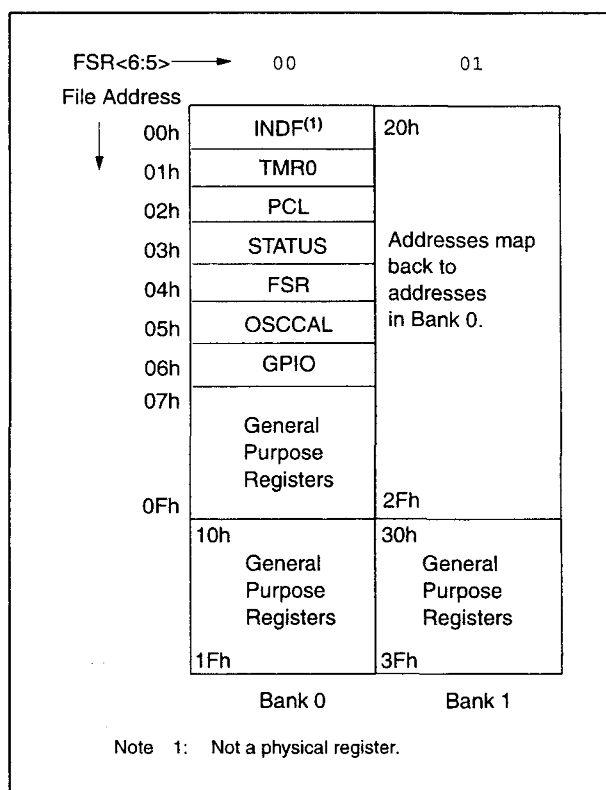
bruikt voor data en besturingsinformatie onder commando van de instructies.
Bij de PIC12C508 bestaat de register-file uit 7 speciale functie-registers en 25 algemene registers (figuur 7/6.5.5-8).
Bij de PIC12C509 is de register-file samengesteld uit 7 speciale functie-registers, 25 algemene registers plus 16 algemene registers die door middel van banking geadresseerd kunnen worden (figuur 7/6.5.5-9).

General Purpose Register File
De algemene register-file kan direct of indirect worden geadresseerd via het file-select register FSR.

Special Function Registers
De registers voor de speciale functies (SFR's) worden door de CPU en perifere functies gebruikt om de werking van de schakeling te regelen (zie tabel 7/6.5.5-2).

Status Register
Dit register bevat de rekenkundige toestand van de ALU, de RESET status en het pagina-preselect-bit voor programmageheugens die groter zijn dan 512 woorden. Het statusregister kan de bestemming zijn van alle instructies, net als elk andere register.
Als het status-register het doel is van een instructie die invloed heeft op de Z, DC of C bits, is het schrijven naar deze drie bits gesperd. Deze bits worden geset of gecleared naar gelang de logika. Bovendien zijn de $\overline{TO}$ en $\overline{PD}$ bits niet beschrijfbaar. Daardoor kan het resultaat van een instructie met het statusregister anders uitpakken dan de bedoeling was.
Met CLRF STATUS worden bijvoorbeeld de bovenste drie bits gecleared en het Z-bit geset. Het statusregister blijft hierdoor achter als 000u u1uu (u = onveranderd).
Het wordt daarom aanbevolen alleen BCF, BSF en MOVWF instructies te gebruiken om het statusregister te veranderen, omdat deze instructies niet de Z, DC of C bits beïnvloeden. Voor andere instructies die wel de STATUS-bits beïnvloeden wordt verwezen naar de instructieset.

Option Register
Het optie-register is een 8 bit breed, write-only register dat allerlei besturingsbits bevat om de TMR0/WDT prescaler en Timer0 te configureren (figuur 7/6.5.5-11). Door uitvoering van de OPTION instructie wordt de inhoud van het W-register overgebracht naar het OPTION register. Door een RESET worden de OPTION-bits <7:0> op "1" gezet (alle bits).



Figuur 7/6.5.5-9: Register-file van de PIC12C509.

6.5 PIC-typen 8 bit microcontrollers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on $\overline{MCLR}$ and WDT Reset	Value on Wake-up on Pin Change
N/A	TRIS	I/O control registers								--11 1111	--11 1111	--11 1111
N/A	OPTION	Contains control bits to configure Timer0, Timer0/WDT prescaler, wake-up on change, and weak pull-ups								1111 1111	1111 1111	1111 1111
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								xxxx xxxx	uuuu uuuu	uuuu uuuu
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu	uuuu uuuu
02h[1]	PCL	Low order 8 bits of PC								1111 1111	1111 1111	1111 1111
03h	STATUS	GPWUF	—	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu	100q quuu
04h	FSR (12C508)	Indirect data memory address pointer								111x xxxx	111u uuuu	111u uuuu
04h	FSR (12C509)	Indirect data memory address pointer								110x xxxx	11uu uuuu	11uu uuuu
05h	OSCCAL	CAL7	CAL6	CAL5	CAL4	—	—	—	—	0111 ----	uuuu ----	uuuu ----
06h	GPIO	—	—	GP5	GP4	GP3	GP2	GP1	GP0	--xx xxxx	--uu uuuu	--uu uuuu

Legend: Shaded boxes = unimplemented or unused, – = unimplemented, read as '0' (if applicable)
x = unknown, u = unchanged, q = see the tables in Section 7.7 for possible values.

Note 1: The upper byte of the Program Counter is not directly accessible. See Section 4.5 for an explanation of how to access these bits.

Tabel 7/6.5.5-2: **Overzicht van de Special Function Registers (SFR).**

6.5 PIC-typen 8 bit microcontrollers

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
GPWUF	—	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7	6	5	4	3	2	1	bit0

R = Readable bit
W = Writable bit
- n = Value at POR reset

bit 7: **GPWUF**: GPIO reset bit
1 = Reset due to wake-up from SLEEP on pin change
0 = After power up or other reset

bit 6: **Unimplemented**

bit 5: **PA0**: Program page preselect bits
1 = Page 1 (200h - 3FFh) - PIC12C509
0 = Page 0 (000h - 1FFh) - PIC12C508 and PIC12C509
Each page is 512 bytes.
Using the PA0 bit as a general purpose read/write bit in devices which do not use it for program page preselect is not recommended since this may affect upward compatibility with future products.

bit 4: **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3: **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2: **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

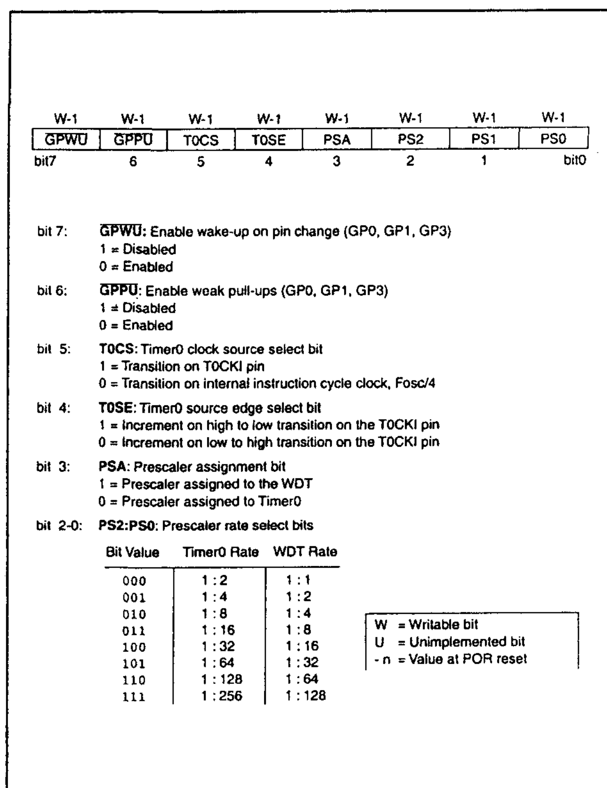
bit 1: **DC**: Digit carry/$\overline{\text{borrow}}$ bit (for ADDWF and SUBWF instructions)
ADDWF
1 = A carry from the 4th low order bit of the result occurred
0 = A carry from the 4th low order bit of the result did not occur
SUBWF
1 = A borrow from the 4th low order bit of the result did not occur
0 = A borrow from the 4th low order bit of the result occurred

bit 0: **C**: Carry/$\overline{\text{borrow}}$ bit (for ADDWF, SUBWF and RRF, RLF instructions)

ADDWF	**SUBWF**	**RRF or RLF**
1 = A carry occurred	1 = A borrow did not occur	Load bit with LSB or MSB, respectively
0 = A carry did not occur	0 = A borrow occurred	

Figuur 7/6.5.5-10: Het Statusregister (adres 03h).

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-11: Het Option-register.

Opmerkingen

- Als het TRIS-bit op "0" is gezet, zijn de "wake-up on change" en "pull-up" functies voor die pen uitgeschakeld. Met andere woorden: TRIS gaat boven OPTION control van $\overline{\text{GPPU}}$ en $\overline{\text{GPWU}}$.
- Als het TOCS-bit op "1" is gezet, wordt GP2 een ingang, ook al is TRIS GP2="0".

Program Counter

Als een programma-instructie wordt uitgevoerd, zal de program-counter (PC) het adres van de volgende, uit te voeren instructie bevatten. Bij elke instructie-cyclus wordt de PC-waarde met één verhoogd, tenzij een instructie zelf de PC wijzigt.

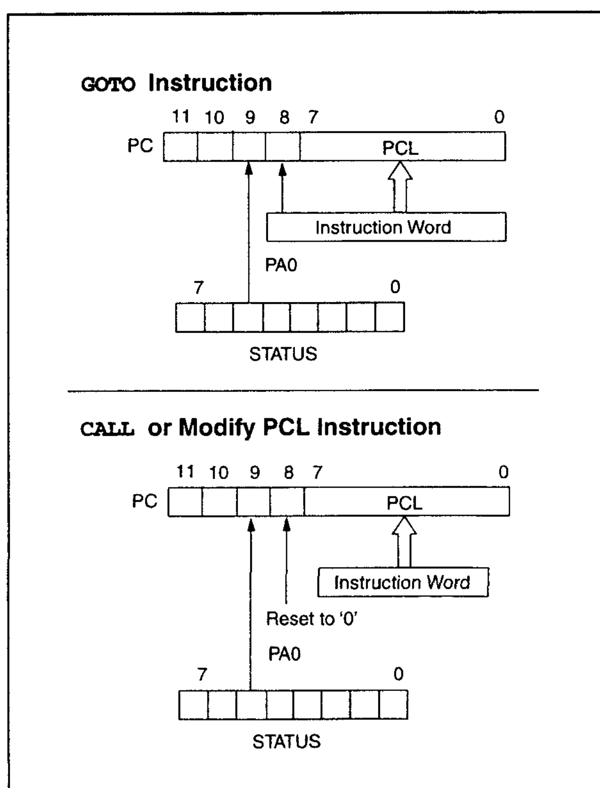
Voor een GOTO-instructie worden de bits 0 tot en met 8 geleverd door het GOTO instructiewoord. De PC-latch (PCL) wordt gemapped naar PC<7:0>. Bit 5 van het STATUS-register levert pagina-informatie aan bit 9 van de PC (figuur 7/6.5.5-12).

Voor een CALL-instructie of een andere instructie waarbij de PCL het doel is, worden de bits 0 tot en met 7 van de PC weer geleverd door het instructiewoord. PC<8> komt hierbij echter niet uit het instructiewoord, maar wordt altijd gecleared.

Tot de instructies waarbij de PCL de bestemming is, of Modify PCL-instructies behoren MOVWF PC, ADDWF PC en BSF PC,5.

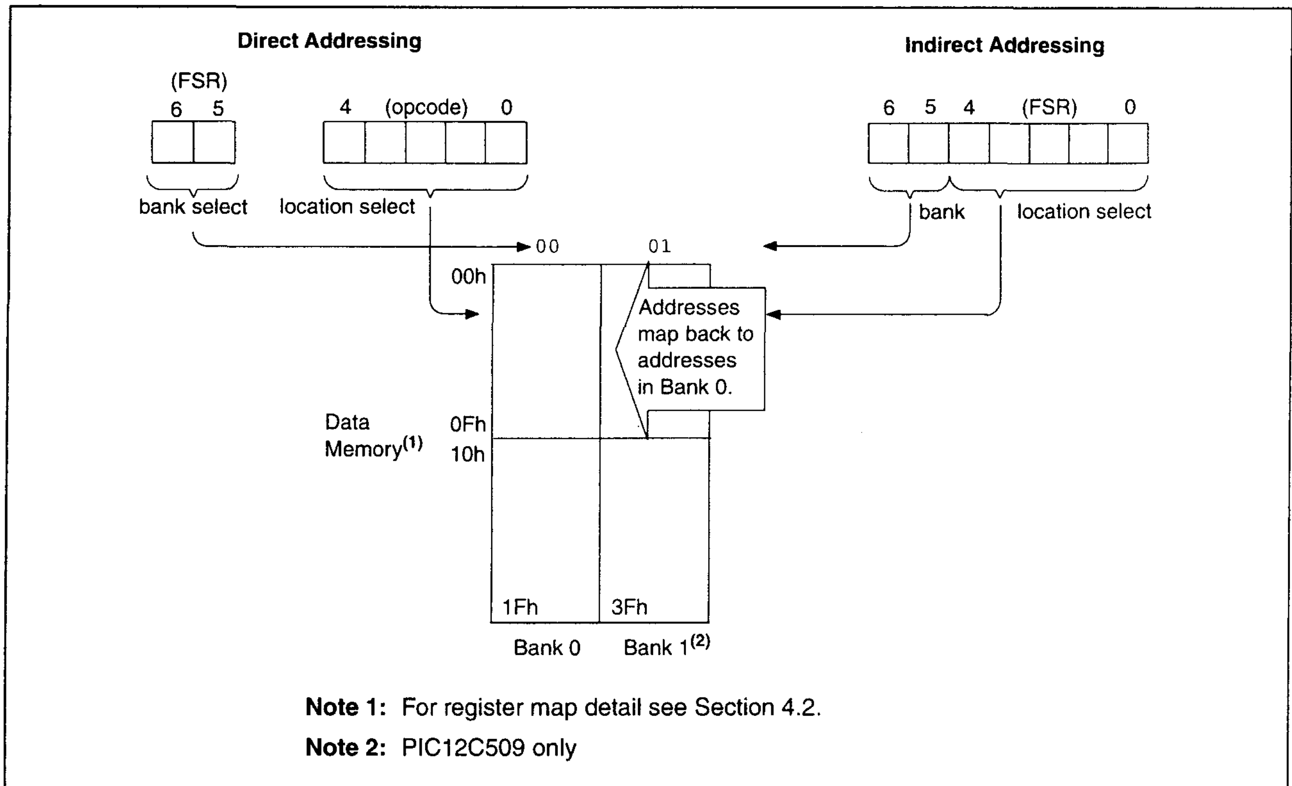
Opmerking

Omdat PC<8> (bit 8 van de PC) in de CALL-instructie of een andere Modify PCL-instructie wordt gecleared, zijn alle subroutine-call's of berekende jumps beperkt tot de eerste 256 locaties van een willekeurige programma-geheugenpagina (512 woorden lang).



Figuur 7/6.5.5-12: Het laden van PC branch-instructies bij de PIC12C508/C509.

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-13: Directe/indirecte adressering.

De effecten van RESET

Na een RESET wordt de Program Counter geset, hetgeen betekent dat de PC de laatste locatie op de laatste pagina adresseert, dat wil zeggen: de oscillator calibratie instructie. Na uitvoering van MOVLW XX zal de PC doorrollen naar lokatie 00h en beginnen met het uitvoeren van de gebruikerscode.
Na een RESET worden de registerpagina-presetbits gecleared, zodat pagina 0 wordt gekozen.
Hierdoor zal het programma door een GOTO-instructie na een RESET automatisch naar pagina 0 springen.

Stack

De PIC12C5xx microcontrollers hebben een 12 bit brede hardware push/pop stack. Een CALL-instructie zal de huidige waarde van stack 1 naar stack 2 overbrengen (push) en daarna de huidige PC-waarde - met één verhoogd - op stack niveau 1 zetten. Als er meer dan twee achtereenvolgende CALL's worden uitgevoerd, zullen alleen de meest recente twee return-adressen worden opgeslagen.
Een RETLW-instructie zal de inhoud van stack niveau 1 in de program-counter laden (pop) en dan de inhoud van stack niveau 2 naar niveau 1 kopiëren. Als er meer dan twee achtereenvolgende RETLW's worden uitgevoerd, zal de stack worden gevuld met het adres dat eerst op niveau 2 was opgeslagen.
Merk op dat het W-register zal worden geladen met de letterlijke waarde die door de instructie wordt gespecificeerd.
Dit is vooral bruikbaar voor de implementatie van data look-up tabellen in het programmageheugen.

Indirecte data adressering, INDF en FSR Registers

Het INDF register is geen fysiek register. Adresseren van INDF betekent eigenlijk: het adresseren van het register waarvan het

adres in het FSR-register staat (FSR is een pointer). Dit is indirect adresseren. Het FSR is een 5 bit breed register dat samen met het INDF-register wordt gebruikt om data-geheugen gebieden indirect te adresseren (zie figuur 7/6.5.5-13).
De FSR<4:0> bits worden gebruikt voor het selecteren van de data-geheugen adressen 00h tot en met 1Fh.

- PIC12C508:
  gebruikt geen "banking", zodat FSR<6:5> niet geïmplementeerd zijn en als "1" worden gelezen.
- PIC12C509:
  gebruikt FSR<5> om te selecteren tussen bank 0 en bank 1. FSR<6> is niet geïmplementeerd en wordt als "1" gelezen.

```
          movlw    0x10     ;initialize pointer
          movwf    FSR      ; to RAM
NEXT      clrf     INDF     ;clear INDF register
          incf     FSR,F    ;inc pointer
          btfsc    FSR,4    ;all done?
          goto     NEXT     ;NO, clear next
CONTINUE
          :                 ;YES, continue
```

Figuur 7/6.5.5-14: Voorbeeld 2: het clearen van RAM, waarbij gebruik wordt gemaakt van indirecte adressering.

Voorbeelden

Voorbeeld 1: Indirecte adressering

- Register-file 07 bevat de waarde 10h
- Register-file 08 bevat de waarde 0Ah
- Laad de waarde 07 in het FSR-register
- Uitlezen van het INDF-register levert de waarde 10h op
- Verhoog de waarde van het FSR-register met één (FSR = 08)
- Uitlezen van het INDF-register levert de waarde 0Ah op.

Het indirect uitlezen van INDF zelf (FSR = 0) zal ooh opleveren. Indirect schrijven naar het INDF-register resulteert in een no-operation (hoewel dit invloed kan hebben op STATUS-bits).
In voorbeeld 2 (figuur 7/6.5.5-14) is een eenvoudig programma te zien waarmee de RAM-lokaties 10h tot en met 1Fh worden gecleared, waarbij gebruik wordt gemaakt van indirecte adressering.

I/O-poort

Net als bij andere registers kan het I/O-register worden uitgelezen en beschreven onder programma-besturing. Met leesinstructies (zoals MOVF GPIO, W) worden echter altijd de I/O-pennen uitgelezen, ongeacht de input/output modes ervan. Na een RESET zijn alle I/O-poorten gedefinieerd als ingang (hoog-impedant) aangezien de I/O besturingsregisters allemaal gezet zijn.

GPIO

GPIO is een 8 bit I/O-register. Alleen de laagste 6 bits worden gebruikt (GP5:GP0). De bits 7 en 6 zijn niet geïmplementeerd en worden als "0" uitgelezen.
Merk op dat GP3 alleen ingang is. Het configuratiewoord kan verschillende I/O's alternatieve functies geven. Als ze alternatieve functies hebben, zullen de pennen als "0" worden uitgelezen. Aan de pennen GP0, GP1 en GP3 kunnen zwakke optrek- en "wake-up on change" functies worden toegewezen.
Deze functies kunnen niet per pen worden geselecteerd. Als pen 4 als $\overline{MCLR}$ is geconfigureerd, staat de zwakke optrekking altijd aan en wake-up on change is dan voor deze pen niet mogelijk.

TRIS Register

Het output-driver besturingsregister wordt geladen met de inhoud van het W-register bij de uitvoering van de TRIS f instructie. Een "1" van een TRIS-registerbit zet de overeenkomstige uitgangsdriver in een hoog-impedante toestand.
Een "0" zet de inhoud van de uitgangs data-latch op de geselecteerde pennen en geeft de uitgangsbuffer vrij.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on $\overline{MCLR}$ and WDT Reset	Value on Wake-up on Pin Change
N/A	TRIS	I/O control registers								--11 1111	--11 1111	--11 1111
N/A	OPTION	$\overline{GPWU}$	$\overline{GPPU}$	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	1111 1111
03H	STATUS	GPWUF	—	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu	100q quuu
06h	GPIO	—	—	GP5	GP4	GP3	GP2	GP1	GP0	--xx xxxx	--uu uuuu	--uu uuuu

Legend: Shaded cells not used by Port Registers, read as '0', — = unimplemented, read as '0', x = unknown, u = unchanged, q = see tables in Section 7.7 for possible values.

Tabel 7/6.5.5-3: Overzicht van de poort-registers.

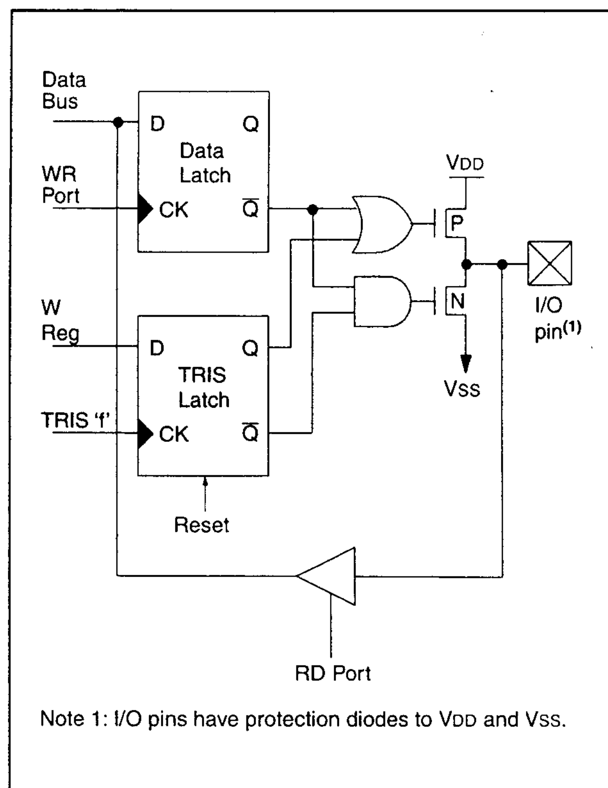
De uitzonderingen zijn GP3 die altijd ingang is en GP2 die bestuurd kan worden door het option-register.
De TRIS-registers zijn "write-only" en worden na een RESET geset (output drivers disabled).

Opmerking

Door een lees-operatie van de poorten worden de pennen en NIET de uitgangs-datalatches uitgelezen. Dus als een uitgangsdriver van een pen is vrijgegeven (enabled) en HOOG wordt gestuurd, maar door het externe systeem LAAG wordt gehouden, zal een lees-operatie van deze poort aangeven dat de pen LAAG is.

I/O interfacing

In figuur 7/6.5.5-15 is de vervangingsschakeling van een I/O-poort te zien. Alle poorten (op GP3 na) kunnen voor ingangs- en uitgangs-operaties worden gebruikt. Voor ingangs-operaties zijn deze poorten niet-latchend. Een ingangssignaal moet dus aanwezig blijven totdat het is ingelezen door een input-instructie (bijvoorbeeld MOVF GPIO, W). De uitgangen zijn gelatched en blijven onveranderd totdat de uitgangslatch is overschreven.
Om een poort-pen als uitgang te gebruiken moet het overeenkomstig direction-control bit in TRIS worden gecleared (= "0"). Voor gebruik als ingang moet het overeenkomstig TRIS-bit "1" zijn. Elke I/O-pen (behalve GP3) kan individueel als ingang of als uitgang worden geprogrammeerd.



Figuur 7/6.5.5-15: Equivalente schakeling van één I/O-pen.

I/O programmeer-overwegingen

Bi-directionele I/O-poorten

Sommige instructies werken intern als lezen, gevolgd door schrijf-operaties. De BCF en BSF instructies lezen bijvoorbeeld de gehele

poort in de CPU, voeren de bit-operatie uit en zetten het resultaat op de uitgang. Voorzichtigheid is geboden als deze instructies worden toegepast op een poort waarvan één of meerdere pennen als in- of uitgang worden gebruikt. Een BSF operatie op bit 5 van GPIO maakt bijvoorbeeld dat alle acht bits van GPIO in de CPU worden ingelezen, waarna bit 5 wordt geset en de GPIO-waarde in de uitgangslatches wordt geschreven. Als een andere bit van GPIO (bijvoorbeeld bit 0) als bi-directionele I/O-pen wordt gebruikt en op dat moment als ingang is gedefinieerd, zal het ingangssignaal op de pen zelf in de CPU worden gelezen en naar de datalatch van deze pen worden geschreven, waarbij de vorige inhoud wordt overschreven. Zolang de pen in de ingangs-mode blijft, is er niets aan de hand. Als bit 0 echter daarna in de uitgangs-mode komt, is de inhoud van de datalatch onbekend.
Voorbeeld 3 (figuur 7/6.5.5-16) laat het effect van twee opvolgende read-modify-write instructies (bijvoorbeeld BCF, BSF, enz.) op een I/O-poort zien.
Een pen met een aktief LAGE of HOGE uitgang moet niet tegelijkertijd door externe schakelingen worden aangedreven omdat daardoor de chip beschadigd kan worden.

Opeenvolgende operaties op I/O-poorten

Het feitelijke schrijven naar een I/O-poort vindt plaats aan het einde van een instructiecyclus.
Bij het lezen ervan moet de data echter al aan het begin van de instructiecyclus aanwezig zijn (zie figuur 7/6.5.5-17). Daarom is voorzichtigheid geboden als na elkaar een schrijf- en een lees-operatie op dezelfde I/O-poort worden uitgevoerd.
De instructie-volgorde dient zo te zijn dat de spanning op de pen stabiel is (afhankelijk van de belasting) vóór de volgende instructie. Voor alle zekerheid kunnen deze instructies worden gescheiden met een NOP of andere instructie die geen invloed heeft op deze I/O-poort.

TIMER0-module en TMR0 register

De Timer0-module heeft de volgende eigenschappen:
- 8 bit timer/counter register, TMR0 lees- en schrijfbaar;
- 8 bit met software programmeerbare prescaler;
- interne of externe clock-select - edge select voor externe clock.

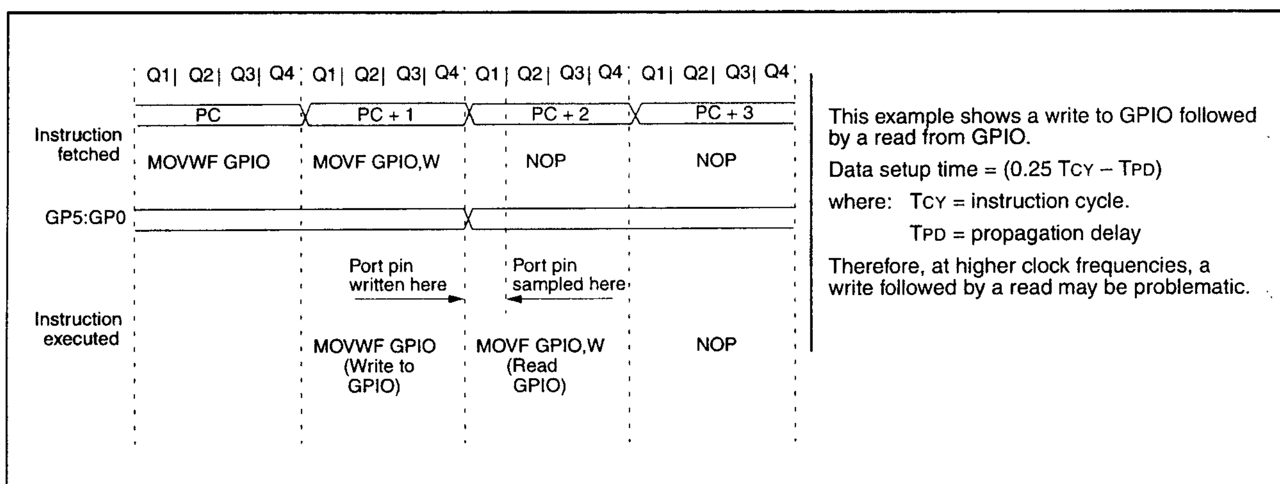
Figuur 7/6.5.5-18 is het vereenvoudigde blokschema van de TMR0-module. De timermode wordt geselecteerd door het T0CS-bit te clearen (OPTION<5>). De TMR0-module zal dan bij elke instructiecyclus met één worden verhoogd (zonder prescaler). Als in het TMR0-register wordt geschreven, wordt het incrementeren gedurende de volgende twee cycli gesperd (figuren 7/6.5.5-19 en -20). De gebruiker kan dit omzeilen door een aangepaste waarde in het TMR0-register te schrijven.
De counter-mode wordt geselecteerd door het T0CS-bit (OPTION<5>) te setten. In deze mode wordt Timer0 op elke stijgende of dalende flank van het signaal op T0CKI met één verhoogd. De flank wordt bepaald door het T0SE-bit (OPTION<4>). Door het T0SE-bit te clearen wordt de stijgende flank geselecteerd.

```
;Initial GPIO Settings
; GPIO<5:3> Inputs
; GPIO<2:0> Outputs
;
;                        GPIO latch  GPIO pins
;                        ----------  ----------
  BCF    GPIO, 5       ;--01 -ppp    --11 pppp
  BCF    GPIO, 4       ;--10 -ppp    --11 pppp
  MOVLW  007h          ;
  TRIS   GPIO          ;--10 -ppp    --11 pppp
;
;Note that the user may have expected the pin
;values to be --00 pppp. The 2nd BCF caused
;GP5 to be latched as the pin value (High).
```

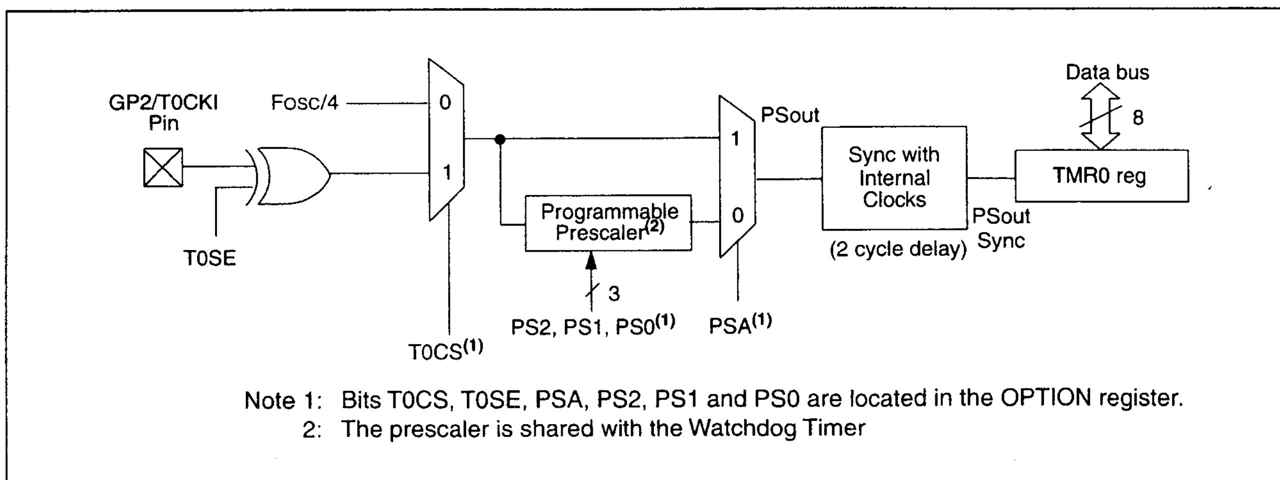
Figuur 7/6.5.5-16: Voorbeeld 3: De effecten van lees-modificeer-schrijf instructies op een I/O-poort.

De prescaler kan worden gebruikt door de Timer0-module of de Watchdog Timer, maar niet door beide tegelijk. De toekenning van de prescaler wordt door software geregeld op het besturingsbit PSA (OPTION<3>). Het clearen van het PSA-bit wijst de prescaler toe aan Timer0. De prescaler kan niet worden uitgelezen of beschreven.

Als de prescaler is toegewezen aan de Timer0-module, kan een prescale-waarde van 1:2, 1:4, ..., 1:256 worden ingesteld.
In tabel 7/6.5.5-4 wordt een overzicht gegeven van de registers die met de Timer0-module te maken hebben.

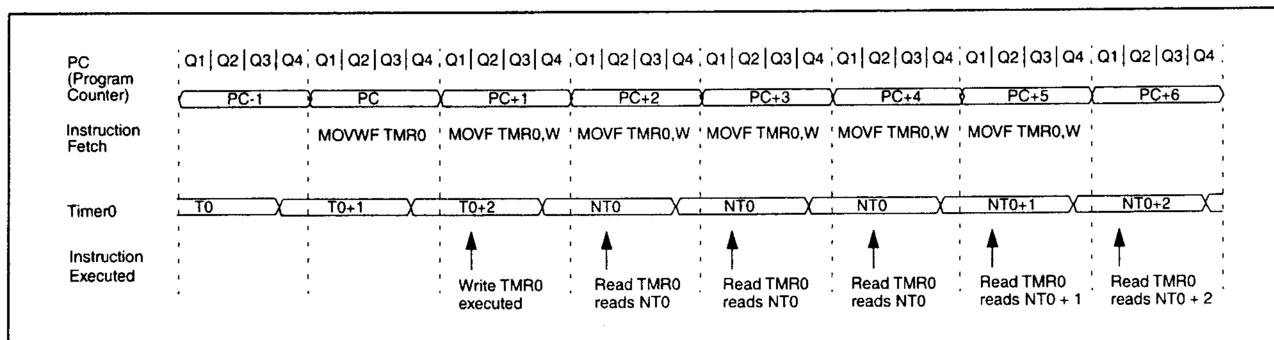


Figuur 7/6.5.5-17: Opeenvolgende I/O-operaties.

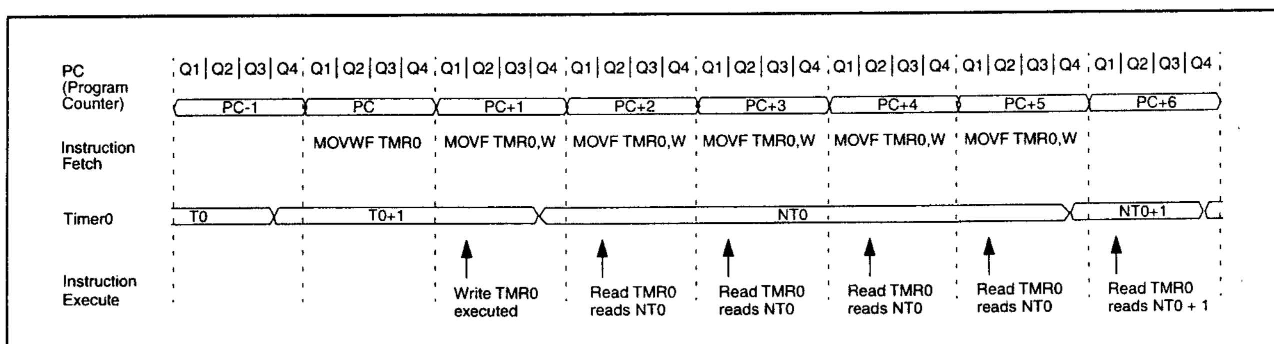


Figuur 7/6.5.5-18: Blokschema van TMR0.

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-19: Timing van Timer0: interne clock/geen prescale.



Figuur 7/6.5.5-20: Timing van Timer0: interne clock/prescale 1:2.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on $\overline{\text{MCLR}}$ and WDT Reset	Value on Wake-up on Pin Change
01h	TMR0	Timer0 - 8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu	uuuu uuuu
N/A	OPTION	$\overline{\text{GPWU}}$	$\overline{\text{GPPU}}$	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	1111 1111
N/A	TRIS	I/O control registers								--11 1111	--11 1111	--11 1111

Legend: Shaded cells not used by Timer0, - = unimplemented, x = unknown, u = unchanged,

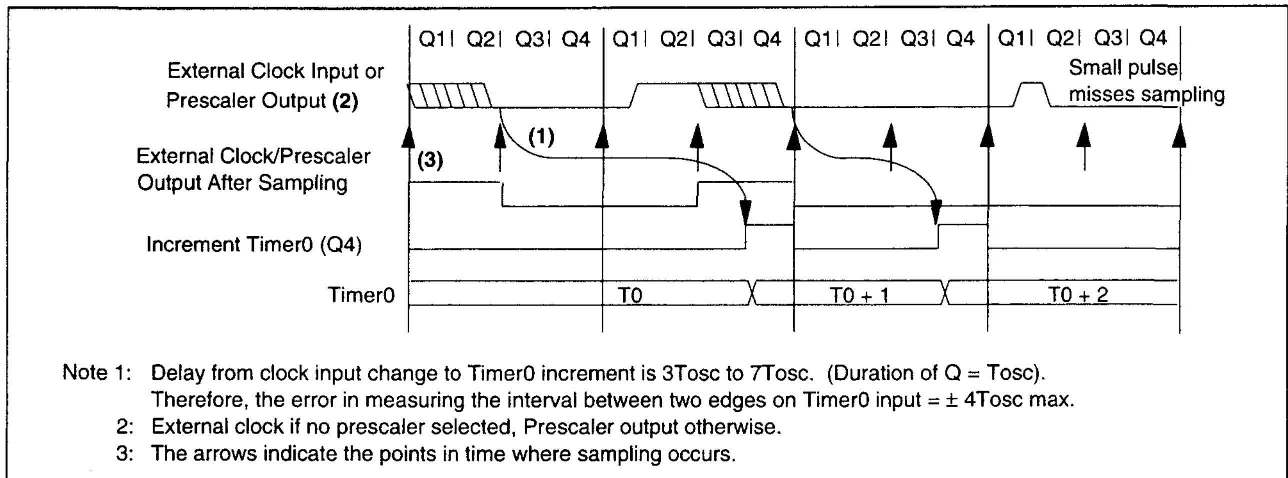
Tabel 7/6.5.5-4: Registers die invloed hebben op Timer0.

Gebruik van Timer0 met een externe clock

Wanneer voor Timer0 een extern clock-signaal wordt gebruikt, worden daaraan eisen gesteld om synchronisatie op de interne phase-clock (T_{OSC}) mogelijk te maken. Ook ontstaat door het incrementeren van Timer0 een vertraging na de synchronisatie.

Externe clock synchronisatie

Wanneer geen prescaler wordt gebruikt is het externe clock-signaal gelijk aan het prescaler uitgangssignaal. Het synchroniseren van T0CKI op de interne phase-clocks wordt verkregen door bemonstering van het prescaler-signaal op de Q2 en Q4 cycli van de interne phase-clocks (figuur 7/6.5.5-21). Daarom moet T0CKI tenminste 2 T_{OSC} (plus een kleine RC-vertraging van 20 ns) HOOG zijn en even lang LAAG.



Figuur 7/6.5.5-21: Timer0 timing met een externe clock.

Als wel een prescaler wordt gebruikt, wordt het externe clock-signaal gedeeld door de prescaler (een asynchrone ripple-counter) om het prescaler-uitgangssignaal symmetrisch te maken. Om de externe clock geschikt te maken voor bemonstering, moet rekening worden gehouden met de ripple-counter. Het is daarvoor nodig dat T0CKI een periode heeft van tenminste 4 * T_{OSC} (+ een RC-vertraging van 40 ns), gedeeld door de prescaler-waarde. De enige eis die aan de HOOG- en LAAG-tijden van T0CKI wordt gesteld is, dat ze groter zijn dan de minimale pulsbreedte van 10 ns.

Timer0 increment vertraging

Aangezien het uitgangssignaal van de prescaler wordt gesynchroniseerd met de interne clocks, is er een kleine vertraging tussen de tijd dat de externe clock-flank optreedt en de tijd dat de Timer0-module echt incrementeert (zie ook figuur 7/6.5.5-21).

Option-register effect op GP2 TRIS

Als het Option-register is ingesteld om Timer0 van de pen te lezen, wordt deze poort een uitgang, ongeacht de instelling in het TRIS-register.

Prescaler

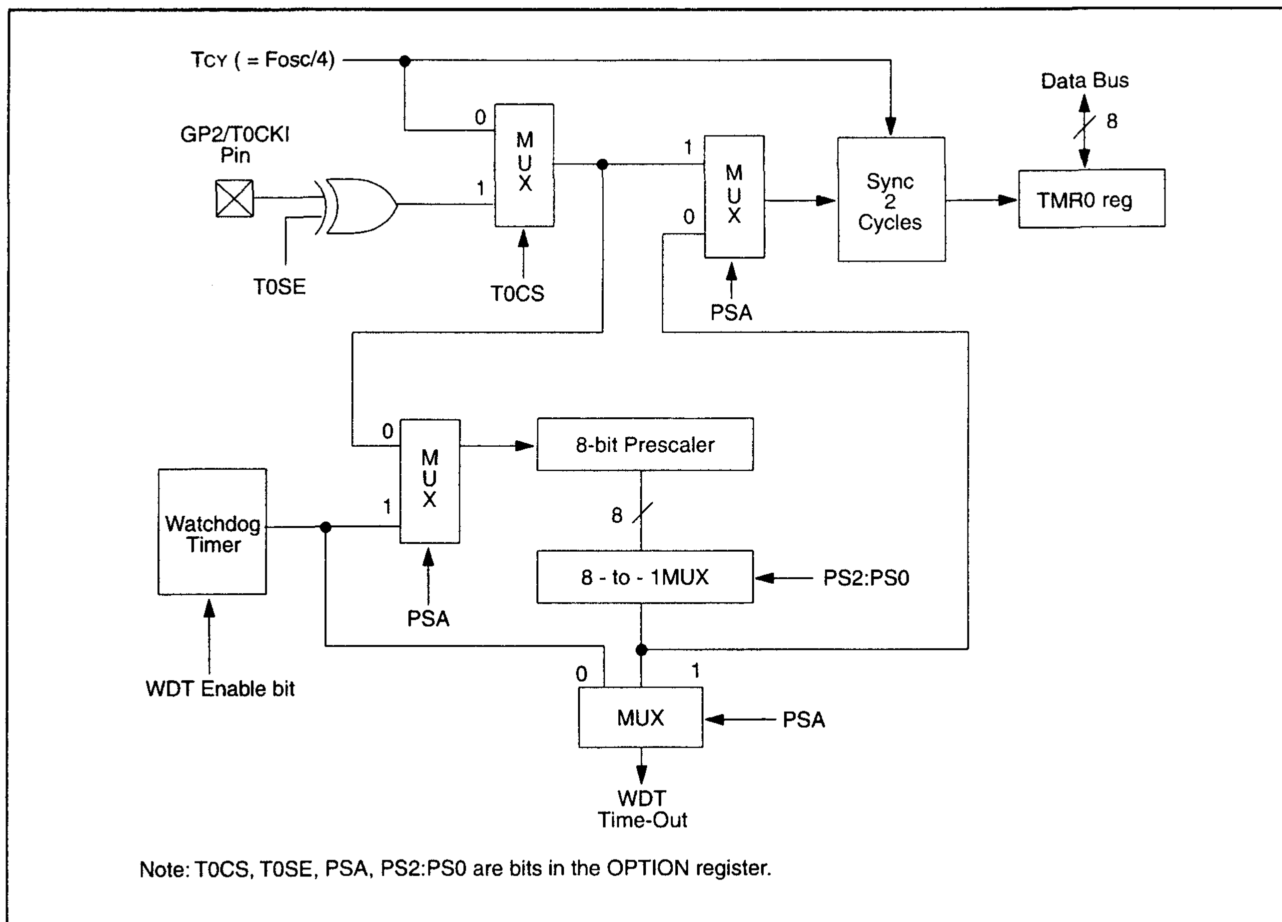
Er is een 8 bit teller beschikbaar als prescaler voor de Timer0-module of als postscaler voor de Watchdog Timer (WDT). Voor het gemak wordt deze teller "prescaler" genoemd. Omdat er slechts één prescaler is, kan deze worden toegewezen aan de Timer0-module of aan de Watchdog Timer, maar niet aan beide tegelijk.
De PSA en PS2:PS0 bits (OPTION<3:0>) bepalen de toewijzing van de prescaler en de prescale-verhouding.
Bij toewijzing aan de Timer0-module zullen alle instructies die naar het Timer0-register schrijven (zoals CLRF 1, MOVWF 1, BSF 1,x...enz.) de prescaler clearen. Bij toewijzing aan de WDT cleart een CLRWDT-instructie de prescaler tegelijk met de Watchdog Timer. De prescaler kan niet worden uitgelezen of beschreven. Na een RESET bevat de prescaler alleen "nullen".

Veranderen van prescaler-toewijzing

De toewijzing van de prescaler gebeurt uitsluitend onder software-besturing en kan dus "on the fly" tijdens het uitvoeren van een programma worden gewijzigd.
Om een onbedoelde RESET te voorkomen moet de in figuur 7/6.5.5-23 vermelde instructie-volgorde van voorbeeld 4 worden uitgevoerd bij het overschakelen van de prescaler tussen Timer0 en WDT.
Om de prescaler van WDT over te laten schakelen naar Timer0 moet de volgorde van voorbeeld 5 (figuur 7/6.5.5-24) worden toe-

6.5 PIC-typen 8 bit microcontrollers

gepast (zelfs als de WDT is uitgeschakeld). Let op dat vóór het omschakelen van de prescaler een CLRWDT instructie moet worden uitgevoerd.



Figuur 7/6.5.5-22: Blokschema van de Timer0/WDT prescaler.

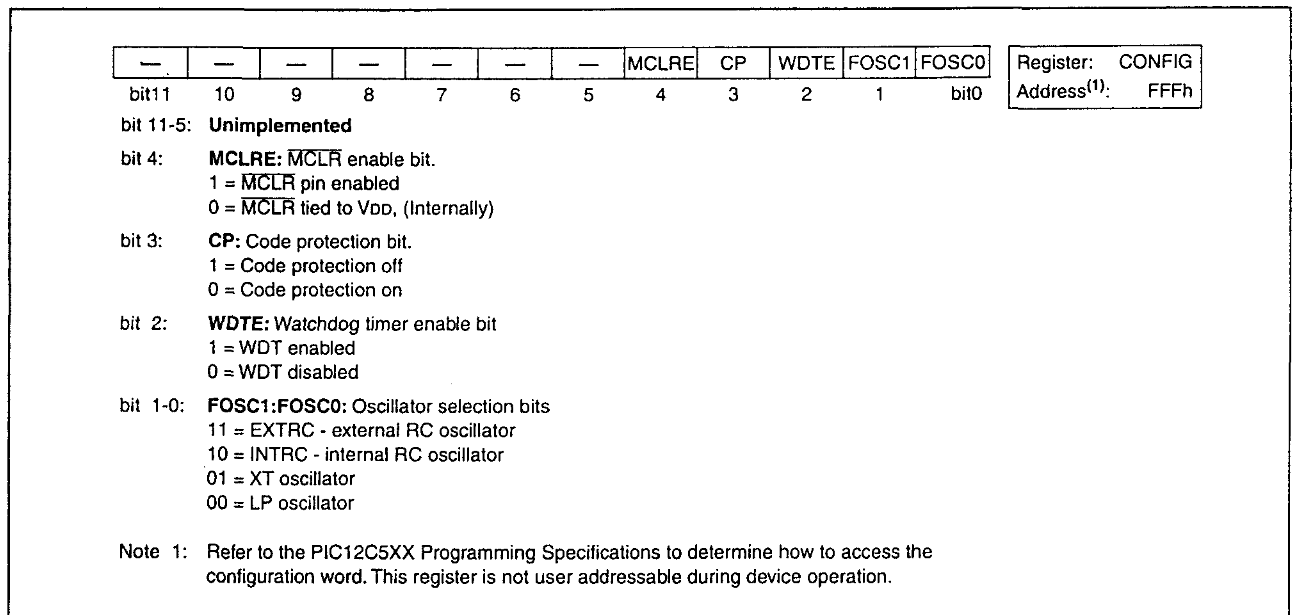
```
1.CLRWDT                ;Clear WDT
2.CLRF   TMR0           ;Clear TMR0 & Prescaler
3.MOVLW  '00xx1111'b; ;These 3 lines (5, 6, 7)
4.OPTION                ; are required only if
                        ; desired
5.CLRWDT                ;PS<2:0> are 000 or 001
6.MOVLW  '00xx1xxx'b    ;Set Postscaler to
7.OPTION                ; desired WDT rate
```

Figuur 7/6.5.5-23: Voorbeeld 4: Overschakelen van de prescaler van Timer0 naar WDT.

```
CLRWDT                        ;Clear WDT and
                              ;prescaler
MOVLW    'xxxx0xxx'           ;Select TMR0, new
                              ;prescale value and
                              ;clock source
OPTION
```

Figuur 7/6.5.5-24: Voorbeeld 4: Prescaler-wisseling van WDT naar Timer0.

6.5 PIC-typen 8 bit microcontrollers



—	—	—	—	—	—	—	MCLRE	CP	WDTE	FOSC1	FOSC0
bit11	10	9	8	7	6	5	4	3	2	1	bit0

Register: CONFIG
Address[1]: FFFh

bit 11-5: **Unimplemented**

bit 4: **MCLRE:** $\overline{\text{MCLR}}$ enable bit.
1 = $\overline{\text{MCLR}}$ pin enabled
0 = $\overline{\text{MCLR}}$ tied to V_{DD}, (Internally)

bit 3: **CP:** Code protection bit.
1 = Code protection off
0 = Code protection on

bit 2: **WDTE:** Watchdog timer enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0: **FOSC1:FOSC0:** Oscillator selection bits
11 = EXTRC - external RC oscillator
10 = INTRC - internal RC oscillator
01 = XT oscillator
00 = LP oscillator

Note 1: Refer to the PIC12C5XX Programming Specifications to determine how to access the configuration word. This register is not user addressable during device operation.

Figuur 7/6.5.5-25: Het configuratiewoord van de PIC126C5xx.

Speciale eigenschappen van de CPU

Inleiding

De PIC12C5xx-familie beschikt over een groot aantal speciale voorzieningen om te voldoen aan de eisen die in real-time toepassingen worden gesteld.

De Watchdog Timer kan alleen worden uitgeschakeld met het configuratiebit WDTE en loopt (voor grotere betrouwbaarheid) op zijn eigen RC-oscillator. Bij gebruik van de XT of LP selecteerbare oscillator-opties levert de Device Reset Timer (DRT) een vertraging van 18 ms om de chip gereset te houden totdat de kristal-oscillator stabiel is. Als INTRC of EXTRC wordt gebruikt is er alleen een 18 ms vertraging bij V_{dd} Power-Up. Door deze on-chip timer is voor de meeste toepassingen geen externe reset-schakeling nodig.

De SLEEP mode is een Power-Down mode, waarbij zeer weinig energie wordt verbruikt. Om hieruit te ontwaken zijn een verandering op een ingangspen of een Watchdog Timer time-out voldoende. Ook staan verschillende oscillator-opties ter beschikking om de PIC12C5xx geschikt te maken voor de toepassing, inclusief een interne 4 MHz oscillator. De EXTRC-oscillator optie maakt het systeem goedkoper, terwijl de LP kristal-optie energie bespaart. Door middel van een set configuratiebits kan uit verschillende mogelijkheden worden gekozen.

Configuratiebits

Het configuratiewoord van de PIC12C5xx bestaat uit 5 bits, zie figuur 7/6.5.5-25. Met twee bits kan het type oscillator worden geselecteerd; één bit is de Watchdog Timer enable-bit en één bit is de $\overline{\text{MCLR}}$ enable-bit. Nog een bit is het code-beveiligingsbit. Bij OTP-schakelingen is de oscillator-configuratie al in de fabriek geprogrammeerd en getest.

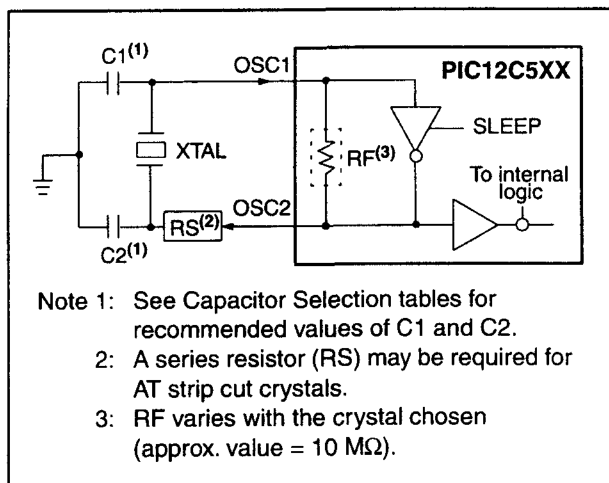
Oscillator-configuraties

De PIC12C5xx kan in vier verschillende oscillator-modes werken. De gebruiker kan met twee configuratiebits (FOSC1:FOSC0) kiezen uit één van de volgende modes:
- LP: Low Power Crystal;
- XT: Crystal/Resonator;

- INTRC: Interne 4 MHz oscillator;
- EXTRC: Externe Resistor/Capacitor.

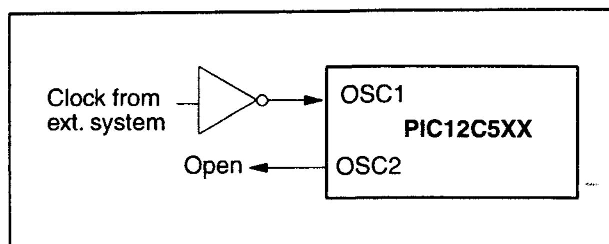
Crystal Oscillator/Ceramic Resonator

In de XT of LP modes is een kristal of een ceramische resonator verbonden met de GP5/OSC1/CLKIN en GP4/OSC2 pennen om het oscilleren te activeren (zie figuur 7/6.5.5-26). Voor de PIC12C5xx is een kristal met parallelle resonantie nodig. In de tabellen 7/6.5.5-5 en -6 zijn de bijbehorende waarden voor de condensatoren opgenomen. In de XT of LP mode is ook een externe clock-aandrijving op GP5/OSC1/CLKIN mogelijk (figuur 7/6.5.5-27).



Note 1: See Capacitor Selection tables for recommended values of C1 and C2.
2: A series resistor (RS) may be required for AT strip cut crystals.
3: RF varies with the crystal chosen (approx. value = 10 MΩ).

Figuur 7/6.5.5-26: Oscillator met kristal of ceramische resonator (XT of LP OSC configuratie).



Figuur 7/6.5.5-27: Oscillator met externe clock-aansturing (XT of LP OSC configuratie).

Osc Type	Resonator Freq	Cap. Range C1	Cap. Range C2
XT	4.0 MHz	30 pF	30 pF

These values are for design guidance only. Since each resonator has its own characteristics, the user should consult the resonator manufacturer for appropriate values of external components.

Tabel 7/6.5.5-5: Condensator-waarden voor een kristal-oscillator.

Osc Type	Resonator Freq	Cap.Range C1	Cap. Range C2
LP	32 kHz(1)	15 pF	15 pF
XT	200 kHz 1 MHz 4 MHz	47-68 pF 15 pF 15 pF	47-68 pF 15 pF 15 pF

Note 1: For VDD > 4.5V, C1 = C2 ≈ 30 pF is recommended.

These values are for design guidance only. Rs may be required in XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components.

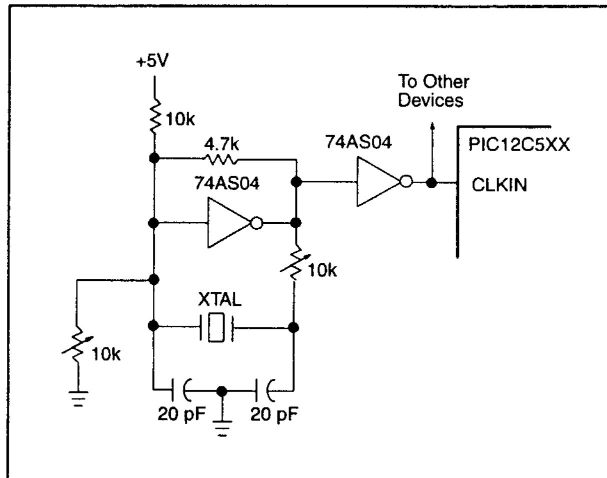
Tabel 7/6.5.5-6: Condensator-waarden voor oscilleren met een ceramische- of kristal-resonator.

Externe kristal-oscillator

Als externe kristal-oscillator kan een kant-en-klare oscillator of een eenvoudige, uit TTL-poorten opgebouwde oscillator worden gebruikt. Kant-en-klare oscillatoren hebben een uitgebreid werkgebied en zijn zeer stabiel, terwijl ook zelf vervaardigde typen met TTL-poorten goed voldoen. Er zijn twee oscillator-schakelingen mogelijk: met parallelle resonantie en met serie resonantie.

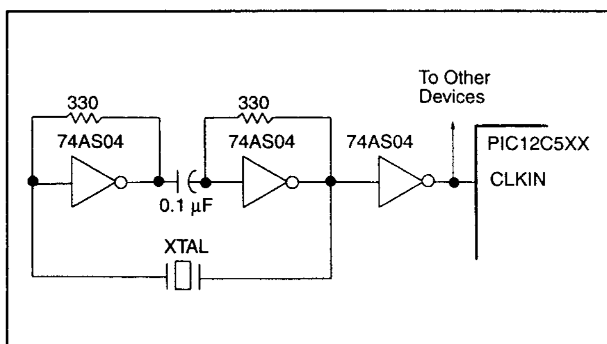
In figuur 7/6.5.5-28 is een parallel oscillerende schakeling te zien, die op de grondfrequentie van het kristal werkt. De 74AS04 inverter zorgt voor de benodigde 180° fasedraaiing, terwijl de 4,7 kΩ weerstand zorgt

voor de stabiliteit (negatieve terugkoppeling). Met de 10 kΩ potentiometer wordt de 74AS04 in het lineaire werkgebied geregeld.



Figuur 7/6.5.5-28: Externe kristal-oscillator met parallelle resonantie.

De in serie oscillerende schakeling in figuur 7/6.5.5-29 werkt ook op de grondfrequentie van het kristal. De inverter zorgt voor een 180° fasedraaiing. De 330 Ω weerstanden leveren de negatieve terugkoppeling om de inverters in hun lineaire werkgebied te houden.



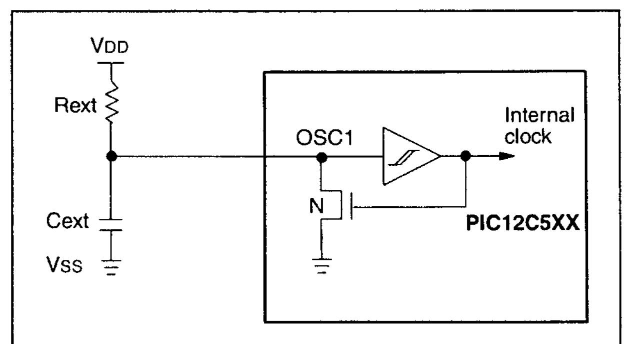
Figuur 7/6.5.5-29: Externe kristal-oscillator met seriële resonantie.

Externe RC-oscillator

Voor tijdsafhankelijke toepassingen is de RC-optie een goede keuze. De RC oscillator-frequentie is een functie van de voedingsspanning, de waarden van weerstand (R_{ext}) en condensator (C_{ext}) en de bedrijfstemperatuur. Bovendien zal de oscillator-frequentie voor elke eenheid afwijken, als gevolg van de normale variaties bij de fabricage. Ook zal de (meetellende) capaciteit van de aansluitdraden per behuizingstype verschillend zijn, vooral bij kleine C_{ext}-waarden. De gebruiker moet ook rekening houden met afwijkingen door de toleranties van de gebruikte R en C componenten.

In figuur 7/6.5.5-30 is een op een PIC12C5xx aangesloten RC-oscillator te zien. Voor waarden van R_{ext}, kleiner dan 2,2 kΩ, kan de oscillator instabiel worden of zelfs helemaal stoppen. Voor zeer grote R_{ext}-waarden (bijvoorbeeld 1 MΩ) wordt de oscillator gevoelig voor storingen, vocht en lekstromen. Er wordt dus aanbevolen R_{ext} tussen 3 kΩ en 100 kΩ te houden.

Hoewel de oscillator ook zonder externe condensator werkt, worden waarden boven 20 pF aanbevolen voor betere stabiliteit en storingsongevoeligheid. Met geen of een zeer kleine externe condensator is de oscillatie-frequentie sterk afhankelijk van veranderingen in de externe capaciteit (bijvoorbeeld van printsporen of aansluitdraden).



Figuur 7/6.5.5-30: RC-oscillator mode.

Interne RC-oscillator

De interne RC-oscillator levert een vaste (nominale) systeemclock van 4 MHz. Daarnaast wordt een calibratie-instructie bovenin het geheugen geprogrammeerd. Deze waarde wordt geprogrammeerd als een MOVLW XX-

instructie, waarbij XX de calibratiewaarde is die bij de reset-vector wordt geplaatst. Hierdoor wordt het W-register na reset geladen met de calibratiewaarde, waarna de PC overgaat naar het gebruikersprogramma op adres 0x000. De gebruiker heeft dan de mogelijkheid om de waarde in het OSCCAL-register (05h) te schrijven of niet.
Wanneer de calibratiewaarde in OSCCAL is geschreven, zal deze de interne oscillator "trimmen". De hoogste 4 bits van het register zijn gereserveerd voor toekomstige calibratie-methoden met langere bit-lengten. Door een grotere waarde op deze plek te schrijven ontstaat een hogere clock-snelheid.

Opmerking
Door het wissen van de inhoud van de schakeling wordt ook de calibratiewaarde voor de interne clock-oscillator gewist.

Reset

Inleiding

De PIC12C5xx maakt onderscheid tussen verschillende soorten reset:

- Power-On Reset (POR);
- $\overline{MCLR}$ reset tijdens normaal bedrijf;
- $\overline{MCLR}$ reset tijdens SLEEP;
- WDT time-out reset tijdens normaal bedrijf;
- WDT time-out reset tijdens SLEEP;
- Wake-up uit SLEEP bij toestandsverandering op een pen.

Sommige registers worden op geen enkele manier gereset. Ze zijn onbekend bij POR en veranderen ook niet door andere reset's. De meeste andere registers worden door POR, $\overline{MCLR}$, een WDT- of Wake-up reset in een "reset toestand" gebracht. Zij worden tijdens SLEEP niet beïnvloed door een WDT-reset of een $\overline{MCLR}$-reset, aangezien deze resets worden beschouwd als een hervatting van de normale werking. De uitzonderingen hierop zijn $\overline{TO}$, $\overline{PD}$ en GPWUF-bits, die afhankelijk van de verschillende resets, worden geset of gecleared. Met deze bits kan in de software worden bepaald om welke soort reset het gaat (zie tabel 7/6.5.5-7 en -8 voor een compleet overzicht van de reset-toestanden van alle registers, respectievelijk de speciale registers).
In figuur 7/6.5.2-32 is een vereenvoudigd blokschema van de on-chip resetschakeling te zien.

Register	Address	Power-on Reset	$\overline{MCLR}$ Reset WDT time-out Wake-up on Pin Change
W	—	qqqq xxxx (1)	qqqq uuuu (1)
INDF	00h	xxxx xxxx	uuuu uuuu
TMR0	01h	xxxx xxxx	uuuu uuuu
PC	02h	1111 1111	1111 1111
STATUS	03h	0001 1xxx	?00? ?uuu (2)
FSR (12C508)	04h	111x xxxx	111u uuuu
FSR (12C509)	04h	110x xxxx	11uu uuuu
OSCCAL	05h	0111 ----	uuuu ----
GPIO	06h	--xx xxxx	--uu uuuu
OPTION	—	1111 1111	1111 1111
TRIS	—	--11 1111	--11 1111

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', ? = value depends on condition.

Note 1: Bits <7:4> of W register contain oscillator calibration values due to MOVLW XX instruction at top of memory.

Note 2: See Table 7-7 for reset value for specific conditions

Tabel 7/6.5.5-7: Overzicht van de reset-condities van de registers.

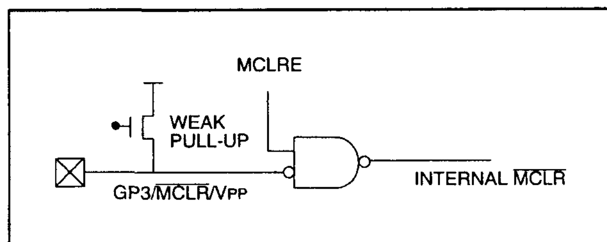
	STATUS Addr: 03h	PCL Addr: 02h
Power on reset	0001 1xxx	1111 1111
$\overline{MCLR}$ reset during normal operation	000u uuuu	1111 1111
$\overline{MCLR}$ reset during SLEEP	0001 0uuuu	1111 1111
WDT reset during SLEEP	0000 0uuu	1111 1111
WDT reset normal operation	0000 1uuu	1111 1111
Wake-up from SLEEP on pin change	1001 0uuu	1111 1111

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0'.

Tabel 7/6.5.5-8: Reset-condities voor speciale registers.

$\overline{MCLR}$ Enable

Dit configuratiebit geeft, wanneer het niet is geprogrammeerd (nog steeds in de "1"-toestand), de externe $\overline{MCLR}$-functie vrij. Als het wel is geprogrammeerd, is de $\overline{MCLR}$-functie verbonden met de interne V_{dd} en wordt de pen toegewezen om als GPIO te werken (zie figuur 7/6.5.5-31).



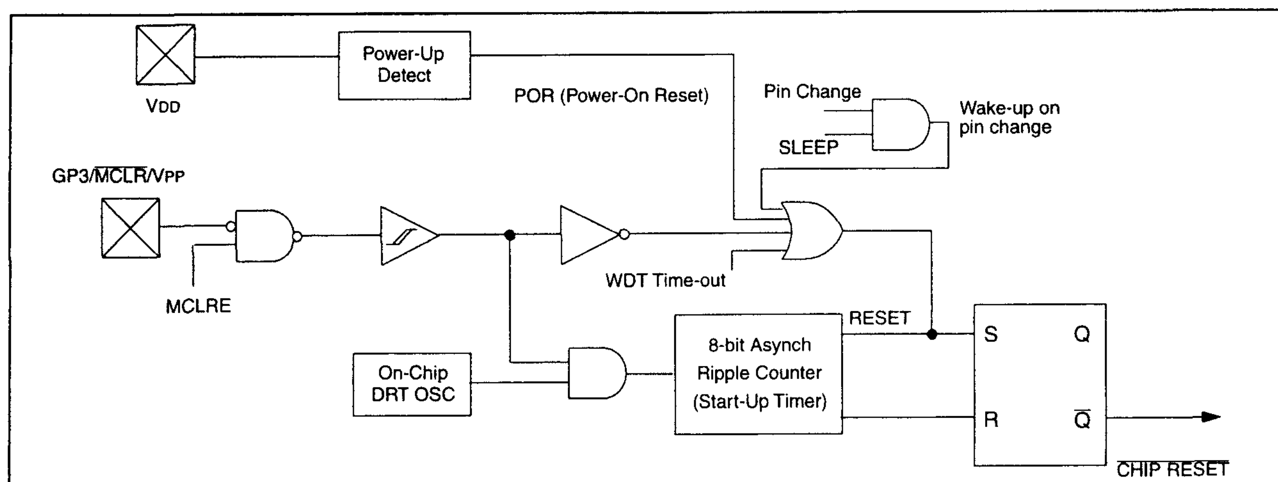
Figuur 7/6.5.5-31: $\overline{MCLR}$ selectie.

Power-On Reset (POR)

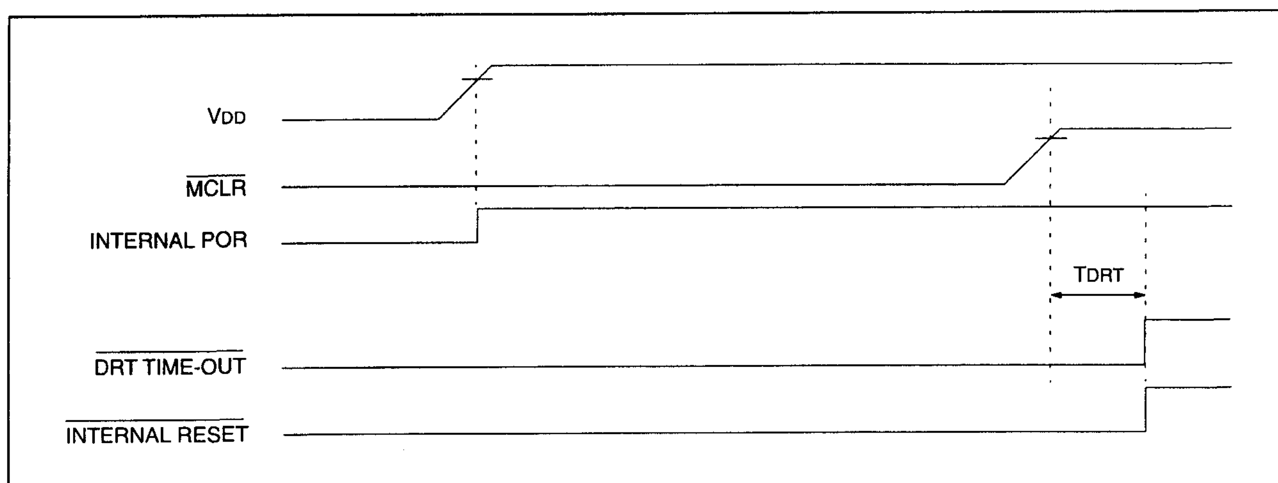
De PIC12C5xx-familie is voorzien van een Power-On Reset schakeling (POR) waarmee voor de meeste power-up situaties een interne chip-reset wordt gegeven. Deze resetpuls wordt op de chip gegenereerd wanneer (in het gebied tussen 1,5 V en 2,1 V) een spanningstoename op V_{dd} wordt gedetecteerd. Om hiervan gebruik te maken moet de GP3/$\overline{MCLER}$/V_{pp}-pen als $\overline{MCLR}$ worden geprogrammeerd en aan V_{DD} worden gelegd of moet de pen als GP3 worden geprogrammeerd.

Een transistor zorgt hierbij voor een zwakke optrekking, waardoor externe RC0-componenten niet nodig zijn. Figuur 7/6.5.5-32 toont een vereenvoudigd blokschema van de on-chip resetschakeling.
De Power-On Reset-schakeling en de Device Reset Timer zijn nauw aan elkaar verwant. Bij power-up wordt de reset-latch geset en de DRT gereset. De DRT-timer begint te tellen zodra hij detecteert dat $\overline{MCLR}$ HOOG is. Na de time-out periode (typisch 18 ms) reset hij de reset-latch en eindigt het on-chip reset-signaal.
Een Power-up voorbeeld, waarbij $\overline{MCLR}$ LAAG wordt gehouden is te zien in figuur 7/6.5.5-33. V_{dd} kan hier stijgen en stabiliseren voordat $\overline{MCLR}$ HOOG gaat. De chip komt T_{DRT} ms na het HOOG gaan van $\overline{MCLR}$ uit de reset.
In figuur 7/6.5.5-34 wordt de Power-On Reset gebruikt. Hierbij zijn $\overline{MCLR}$ en V_{dd} met elkaar verbonden of is de pen geprogrammeerd als GP3. De V_{dd} is stabiel voordat de start-up timer klaar is en de reset werkt prima.
In figuur 7/6.5.5-35 is echter een geval te zien waarbij V_{dd} te langzaam toeneemt. De tijd tussen het punt waarbij de DRT ziet dat $\overline{MCLR}$ HOOG is en het punt waar $\overline{MCLR}$ (en V_{dd}) werkelijk hun volle waarde bereiken duurt hierbij te lang.
In een dergelijke situatie wordt het gebruik van externe RC-schakelingen aanbevolen.

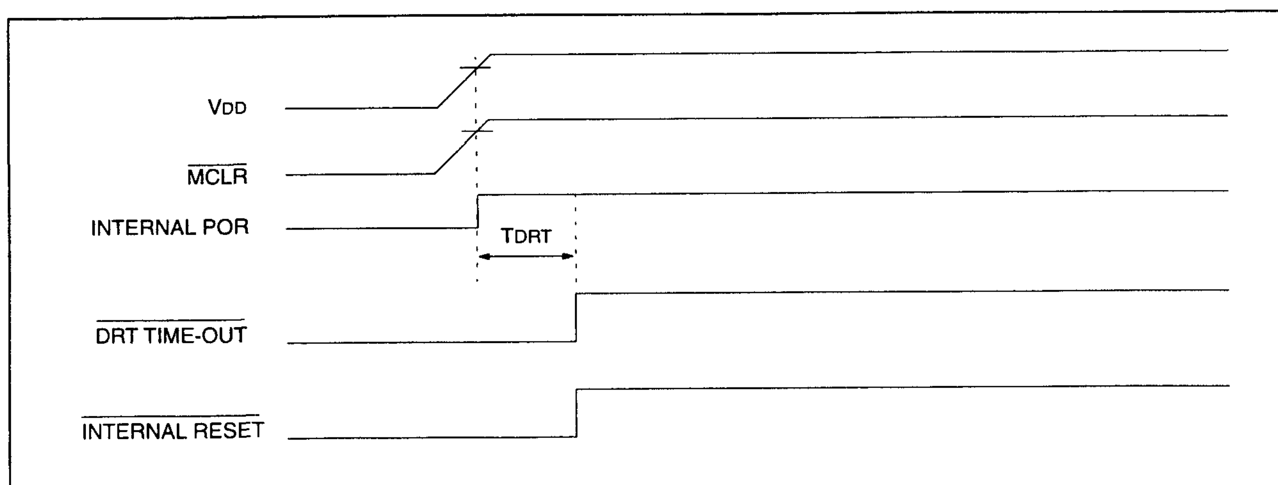
6.5 PIC-typen 8 bit microcontrollers



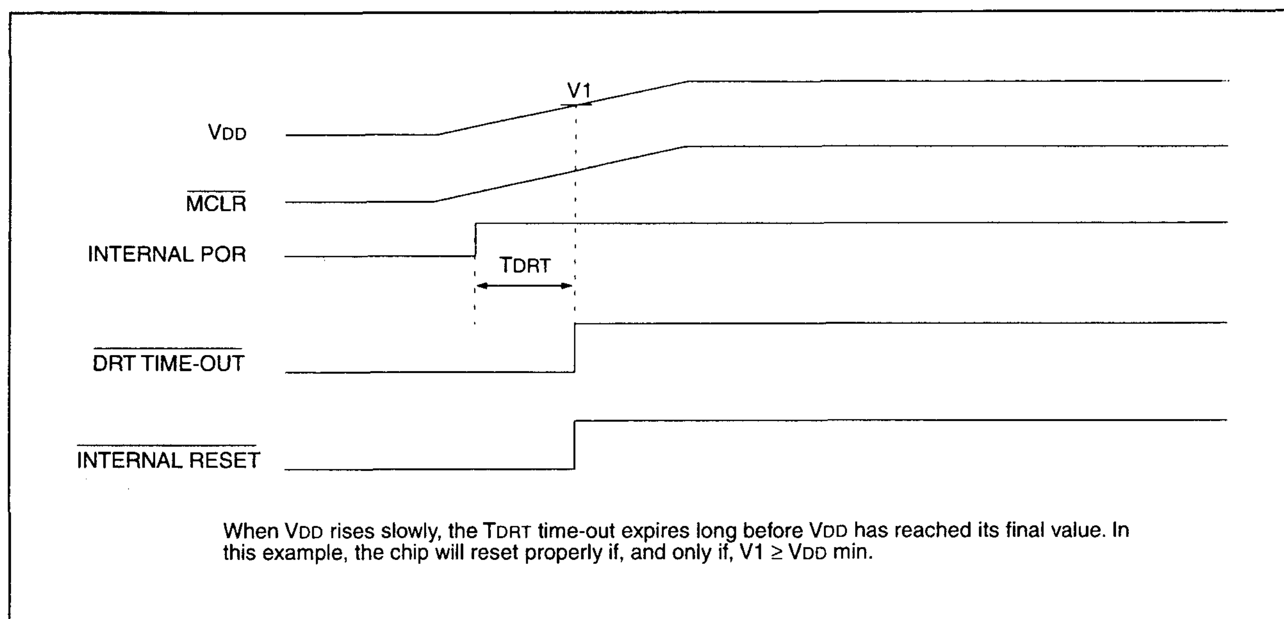
Figuur 7/6.5.5-32: Blokschema van de on-chip reset-schakeling.



Figuur 7/6.5.5-33: Time-out volgorde bij Power-up (waarbij MCLR LAAG wordt getrokken).



Figuur 7/6.5.5-34: Time-out volgorde bij Power-up (MCLR verbonden met VDD): snelle stijging van Vdd.



Figuur 7/6.5.5-35: Time-out volgorde bij Power-up (MCLR verbonden met V_{DD}). Gevaarlijke situatie: langzame stijging van Vdd.

Device Reset Timer (DRT)

In de PIC12C5xx loopt de DRT telkens als de voedingsspanning terugkomt op de schakeling. DRT begint na RESET en varieert, afhankelijk van de gekozen oscillator (zie tabel 7/6.5.5-9). De Device Reset Timer (DRT) levert een vaste nominale time-out van 18 ms na resetten. De DRT werkt met een interne RC-oscillator. De processor wordt RESET gehouden zolang de DRT aktief is. De DRT vertraging maakt dat de V_{dd} boven $V_{dd(min)}$ kan stijgen en dat de oscillator kan stabiliseren.
Wanneer kristallen of ceramische resonatoren worden gebruikt moet na het inschakelen van de voedingsspanning een bepaalde tijd worden gewacht om het oscilleren te laten stabiliseren. De on-chip DRT houdt de PIC12C5xx in een RESET-conditie tot ongeveer 18 ms na het HOOG worden van MCLR. Daardoor zijn meestal geen externe RC-netwerken op de MCLR-ingang nodig en hoeft GP3/MLCR/V_{PP} niet als MCLR te worden geprogrammeerd. Hierdoor gaat geen ruimte verloren en kan de GP3/MLCR/V_{PP}-pen als algemene ingang worden gebruikt.
De Device Reset tijdvertraging zal bij elke chip anders zijn, afhankelijk van V_{DD}, temperatuur en proces-variabelen. De DRT zal ook worden getriggerd bij een Watchdog Timer time-out (alleen in de XT en LP modes). Dit is vooral belangrijk bij toepassingen die de WDT gebruiken om de PIC12C5xx automatisch uit SLEEP te laten ontwaken.

Oscillator Configuration	POR Reset	Subsequent Resets
IntRC & ExtRC	18 ms (typical)	300 µs (typical)
XT & LP	18 ms (typical)	18 ms (typical)

Tabel 7/6.5.5-9: DRT (Device Reset Timer-periode).

6.5 PIC-typen 8 bit microcontrollers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on $\overline{MCLR}$ and WDT Reset	Value on Wake-up on Pin Change
N/A	OPTION	$\overline{GPWU}$	$\overline{GPPU}$	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	1111 1111

Legend: Shaded boxes = Not used by Watchdog Timer, – = unimplemented, read as '0', u = unchanged

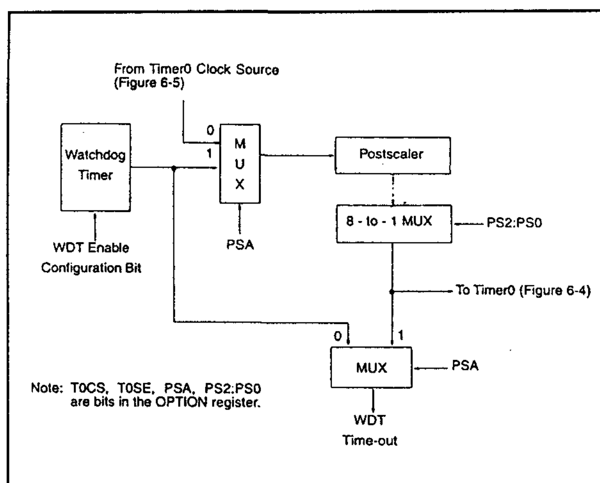
Tabel 7/6.5.5-10: Samenvatting van de registers die betrekking hebben op de Watchdog Timer.

Watchdog Timer (WDT)

De Watchdog Timer (WDT) is een vrijlopende on-chip RC-oscillator waarvoor geen externe componenten nodig zijn. Deze RC-oscillator is gescheiden van de externe RC-oscillator op de GP5/OSC1/CLKIN-pen en de interne 4 MHz oscillator. Dit betekent dat de WDT ook zal lopen als de hoofd processor-clock is gestopt, bijvoorbeeld door een SLEEP-instructie. Tijdens normaal bedrijf of SLEEP genereert een WDT time-out een device-RESET.
De $\overline{TO}$-bit (STATUS) wordt gecleared na een Watchdog Timer reset. De WDT kan permanent worden uitgeschakeld door het configuratiebit WDTE als "0" te programmeren.

WDT-periode

De WDT heeft een nominale time-out periode van 18 ms (zonder prescaler). Als een langere time-out periode gewenst is, kan een prescaler met een deelverhouding van maximaal 1:128 worden toegewezen aan de WDT door in het OPTION-register te schrijven. Met software kan dus een time-out periode van 2,3 s worden gerealiseerd. Deze perioden variëren met de temperatuur, V_{dd} en proces-variabelen. Onder de slechtste omstandigheden (V_{dd} = min., temperatuur = max., max. WDT prescaler) kan het een aantal seconden duren voordat een WDT time-out optreedt.
De CLRWDT-instructie cleart de WDT en de postscaler (als die aan de WDT is toegewezen) en voorkomt dat een device-RESET wordt gegeven.
De SLEEP-instructie reset de WDT en postscaler (als die aan de WDT is toegewezen). Hierdoor onstaat de maximale SLEEP-tijd voordat een WDT wake-up reset optreedt.



Figuur 7/6.5.5-36: Blokschema van de Watchdog Timer.

Time-Out volgorde, Power Down en Wake-up uit SLEEP statusbits ($\overline{TO}$/$\overline{PD}$/GPWUF)

De ($\overline{TO}$, $\overline{PD}$ en GPWUF bits in het STATUS-register kunnen worden getest om te bepalen of een RESET-conditie werd veroorzaakt door een power-up conditie, een $\overline{MCLR}$ of Watchdog Timer (WDT) reset. Deze statusbits worden alleen beïnvloed door gebeurtenissen die in tabel 7/6.5.5-12 zijn vermeld.

Reset na brown-out

Een brown-out is een conditie waarbij de voedingsspanning (V_{dd}) even onder de minimumwaarde komt (maar niet nul wordt) en daarna weer terugkomt. In dit geval dient de schakeling gereset te worden. Om de

PIC12C5xx bij het optreden van een brown-out te resetten kan een externe brown-out beveiliging worden aangebracht, zoals in de figuren 7/6.5.5-37 en -38 is te zien.

GPWUF	$\overline{TO}$	$\overline{PD}$	RESET caused by
0	0	0	WDT wake-up from SLEEP
0	0	1	WDT time-out (not from SLEEP)
0	1	0	$\overline{MCLR}$ wake-up from SLEEP
0	1	1	Power-up
0	u	u	$\overline{MCLR}$ not during SLEEP
1	1	0	Wake-up from SLEEP on pin change

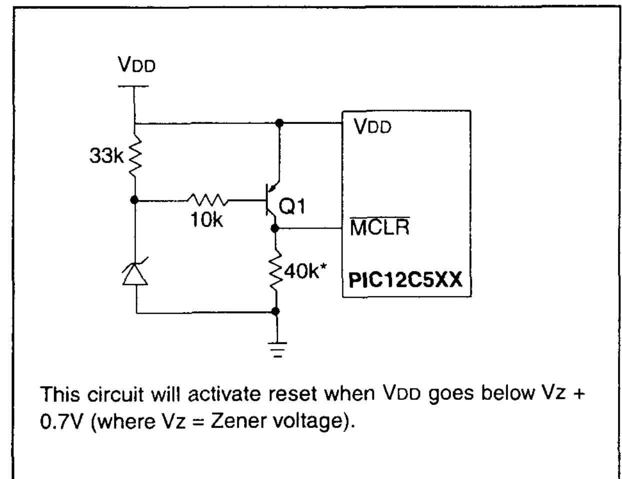
Legend: Legend: u = unchanged
Note 1: The $\overline{TO}$, $\overline{PD}$, and GPWUF bits maintain their status (u) until a reset occurs. A low-pulse on the $\overline{MCLR}$ input does not change the $\overline{TO}$, $\overline{PD}$, and GPWUF status bits.

Tabel 7/6.5.5-11: De status van $\overline{TO}/\overline{PD}$/GPWUF na RESET.

Event	GPWUF	$\overline{TO}$	$\overline{PD}$	Remarks
Power-up	0	1	1	
WDT Time-out	0	0	u	No effect on $\overline{PD}$
SLEEP instruction	u	1	0	
CLRWDT instruction	u	1	1	
Wake-up from SLEEP on pin change	1	1	0	

Legend: u = unchanged
A WDT time-out will occur regardless of the status of the $\overline{TO}$ bit. A SLEEP instruction will be executed, regardless of the status of the $\overline{PD}$ bit.

Tabel 7/6.5.5-12: Gebeurtenissen die effect hebben op de $\overline{TO}/\overline{PD}$ statusbits.



This circuit will activate reset when VDD goes below Vz + 0.7V (where Vz = Zener voltage).

Figuur 7/6.5.5-37: Brown-out beveiligingsschakeling 1.

VDD R1 R2 Q1 40k VDD $\overline{MCLR}$ PIC12C5XX

This brown-out circuit is less expensive, although less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

$$V_{DD} \cdot \frac{R1}{R1 + R2} = 0.7V$$

Figuur 7/6.5.5-38: Brown-out beveiligingsschakeling 2.

Power-Down Mode (SLEEP)

De PIC12C5xx kan op een laag vermogen worden gezet (SLEEP) en daaruit weer terugkomen (Wake-up from SLEEP).

- Sleep

  De Power-Down mode wordt bereikt door een SLEEP-instructie uit te voeren. Een vrijgegeven (enabled) Watchdog Timer

wordt hierdoor gecleared, maar blijft wel werken: het $\overline{TO}$-bit (STATUS) wordt geset, het $\overline{PD}$-bit (STATUS) wordt gecleared en de oscillator-driver wordt uitgeschakeld. De I/O-poorten blijven in de toestand die zij hadden vóór uitvoering van de SLEEP-instructie (HOOG, LAAG of hoogimpedant).
Let op dat de $\overline{MCLR}$-pen niet LAAG gaat, wanneer een WDT time-out een RESET genereert.
Voor het laagste stroomverbruik bij power-down moet de T0CKI-ingang op V_{dd} of V_{ss} staan en de GP3/$\overline{MCLR}$/V_{pp}-pen op logisch HOOG (V_{IHMC}) als de $\overline{MCLR}$ is vrijgegeven.

- Ontwaken uit SLEEP

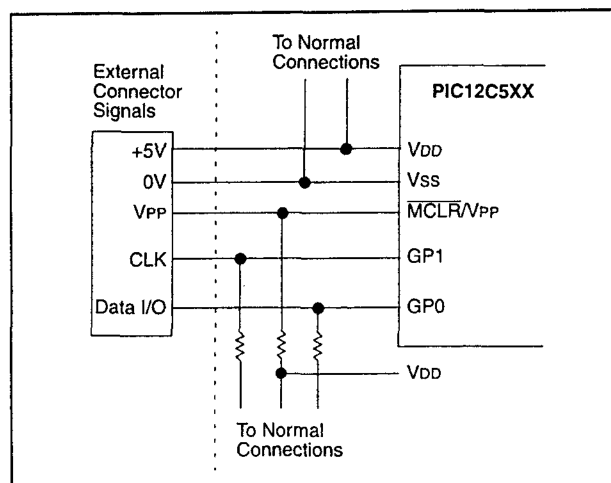
  De PIC12C5xx kan op één van de volgende manieren uit SLEEP ontwaken:
  - Een externe reset op de GP3/$\overline{MCLR}$/V_{pp}-pen als die als $\overline{MCLR}$ werd geconfigureerd.
  - Een Watchdog Timer time-out (als de WDT was vrijgegeven).
  - Een verandering op ingangspen GP0, GP1 of GP3 of GP3/$\overline{MCLR}$/V_{pp} als de "wake-up on change" is vrijgegeven.

  Deze gebeurtenissen veroorzaken een device-reset. De $\overline{TO}$, $\overline{PD}$ en GPWUF-bits kunnen worden gebruikt om de resetoorzaak te detecteren. Het $\overline{TO}$-bit wordt gecleared als een WDT time-out optrad en het ontwaken veroorzaakte. Het $\overline{PD}$-bit, dat bij power-up wordt gezet, wordt bij SLEEP gecleared. Het GPWUF-bit geeft aan dat een toestandsverandering op GP1, GP2 of GP3 optrad (sinds de laatste keer dat een file- of bit-operatie werd uitgevoerd op een GP-poort). De WDT wordt bij het ontwaken uit SLEEP altijd gecleared.

  Opmerking: lees vlak voor het ingaan van de SLEEP-mode de ingangspennen uit. In de SLEEP-mode treedt ontwaken op als de toestand op de pennen verandert. Gebeurt dit en worden de pennen niet uitgelezen voordat weer naar de SLEEP-toestand wordt gegaan, dan treedt onmiddellijk een nieuwe wake-up op, zelfs als daarbij geen verandering op de pennen plaatsvond.

Programma-verificatie/Code-beveiliging
Als het codebeveiligings-bit niet is geprogrammeerd kan het on-chip programmageheugen voor verificatie-doeleinden worden uitgelezen.
De eerste 64 plaatsen kunnen sowieso worden uitgelezen.

ID-locaties
Vier geheugenplaatsen zijn bestemd als ID-locaties waar de gebruiker de checksum of andere code-identificatiegetallen kan opslaan. Deze locaties zijn tijdens normaal bedrijf niet toegankelijk, maar kunnen wel uitgelezen en beschreven worden tijdens program/verify.
Opmerking: gebruik alleen de laagste 4 bits van de ID-lokaties en programmeer de hoogste 8 bits als "0".



Figuur 7/6.5.5-39: Verbindingen bij het in-circuit programmeren.

In-circuit seriële programmering
De PIC12C5xx microcontrollers kunnen serieel worden geprogrammeerd, terwijl ze reeds op de print zijn gesoldeerd.

6.5 PIC-typen 8 bit microcontrollers

Mnemonic, Operands		Description	Cycles	12-Bit Opcode MSb		LSb	Status Affected	Notes
ADDWF	f,d	Add W and f	1	0001	11df	ffff	C,DC,Z	1,2,4
ANDWF	f,d	AND W with f	1	0001	01df	ffff	Z	2,4
CLRF	f	Clear f	1	0000	011f	ffff	Z	4
CLRW	–	Clear W	1	0000	0100	0000	Z	
COMF	f, d	Complement f	1	0010	01df	ffff	Z	
DECF	f, d	Decrement f	1	0000	11df	ffff	Z	2,4
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	0010	11df	ffff	None	2,4
INCF	f, d	Increment f	1	0010	10df	ffff	Z	2,4
INCFSZ	f, d	Increment f, Skip if 0	1(2)	0011	11df	ffff	None	2,4
IORWF	f, d	Inclusive OR W with f	1	0001	00df	ffff	Z	2,4
MOVF	f, d	Move f	1	0010	00df	ffff	Z	2,4
MOVWF	f	Move W to f	1	0000	001f	ffff	None	1,4
NOP	–	No Operation	1	0000	0000	0000	None	
RLF	f, d	Rotate left f through Carry	1	0011	01df	ffff	C	2,4
RRF	f, d	Rotate right f through Carry	1	0011	00df	ffff	C	2,4
SUBWF	f, d	Subtract W from f	1	0000	10df	ffff	C,DC,Z	1,2,4
SWAPF	f, d	Swap f	1	0011	10df	ffff	None	2,4
XORWF	f, d	Exclusive OR W with f	1	0001	10df	ffff	Z	2,4
BIT-ORIENTED FILE REGISTER OPERATIONS								
BCF	f, b	Bit Clear f	1	0100	bbbf	ffff	None	2,4
BSF	f, b	Bit Set f	1	0101	bbbf	ffff	None	2,4
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	0110	bbbf	ffff	None	
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	0111	bbbf	ffff	None	
LITERAL AND CONTROL OPERATIONS								
ANDLW	k	AND literal with W	1	1110	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	1001	kkkk	kkkk	None	1
CLRWDT	k	Clear Watchdog Timer	1	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
GOTO	k	Unconditional branch	2	101k	kkkk	kkkk	None	
IORLW	k	Inclusive OR Literal with W	1	1101	kkkk	kkkk	Z	
MOVLW	k	Move Literal to W	1	1100	kkkk	kkkk	None	
OPTION	–	Load OPTION register	1	0000	0000	0010	None	
RETLW	k	Return, place Literal in W	2	1000	kkkk	kkkk	None	
SLEEP	–	Go into standby mode	1	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	
TRIS	f	Load TRIS register	1	0000	0000	0fff	None	3
XORLW	k	Exclusive OR Literal to W	1	1111	kkkk	kkkk	Z	

Note 1: The 9th bit of the program counter will be forced to a '0' by any instruction that writes to the PC except for GOTO. (Section 4.5)

2: When an I/O register is modified as a function of itself (e.g. `MOVF GPIO, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

3: The instruction `TRIS f`, where f = 6 causes the contents of the W register to be written to the tristate latches of GPIO. A '1' forces the pin to a hi-impedance state and disables the output buffers.

4: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared (if assigned to TMR0).

Tabel 7/6.5.5-13: Samenvatting van de PIC12C5xx instructieset.

Het programmeren vindt eenvoudig plaats met twee lijnen voor clock en data en drie andere lijnen voor voeding, aarde en de programmeerspanning. Dit stelt de gebruikers in staat gedrukte schakelingen met ongeprogrammeerde IC's te vervaardigen.

De micro-controller kan dan vlak voor het transport worden geprogrammeerd, waar-

door steeds de nieuwste firmware beschikbaar is. De schakeling wordt in de program/verify mode geplaatst door de spanning op de $\overline{MCLR}$ (V_{PP})-pen te verhogen van V_{IL} tot V_{IHH} en daarbij de GP1 en GP0 pennen LAAG te houden. Hierdoor wordt GP1 de programmeer-clock en GP0 de programmeer-data. Zowel GP0 als GP1 heeft hierbij een Schmitt-trigger ingang. Na het resetten wordt nu een 6 bit commando aan de schakeling gegeven. Afhankelijk van dit commando (lezen of schrijven) worden 14 bit programmeer-data van of naar de schakeling gevoerd. In figuur 7/6.5.5-39 is een typische in-circuit programmeeopstelling te zien.

Instructieset

Inleiding

Elke PIC12C5xx-instructie is een 12 bit woord, opgedeeld in een OPCODE (die de soort instructie specificeert) en één of meer OPERANDS die de operatie van de instructie verder specificeren. Tabel 7/6.5.5-13 geeft een samenvatting van byte-georiënteerde en bit-georiënteerde instructies en letterlijke en besturings operaties. In tabel 7/6.5.5-14 zijn de opcode field-beschrijvingen te zien.

- Voor byte-georiënteerde instructies stelt "f" een file register-designator voor en "d" een bestemmings- (destination-) designator. De file register-designator wordt gebruikt om één van de 32 file-registers te specificeren die door de instructie worden gebruikt. De destination-designator specificeert waar het resultaat van de operatie naar toe gaat. Als "d" een "0" is, wordt het resultaat in het W-register geplaatst. Als "d" een "1" is, gaat het resultaat naar het in de instructie gespecificeerde file-register.
- Voor bit-georiënteerde instructies is "b" een bit-field designator die het nummer van het bit selecteert dat door de operatie wordt beïnvloed. Hierbij is "f" het nummer van de file waarin het bit zich bevindt.
- Voor letterlijke en besturings operaties, geeft "k" een 8 of 9 bit constante of een letterlijke waarde aan.

Alle instructies worden in één enkele instructiecyclus uitgevoerd, tenzij een conditionele test "waar" is of als de programcounter als gevolg van een instructie is veranderd. In dat geval zijn voor de uitvoering twee instructie-cycli nodig. Een instructie bestaat uit vier oscillator-perioden. Bij een oscillator-frequentie van 4 MHz is de normale uitvoeringstijd van een instructie dus 1 µs.

In tabel 7/6.5.5-15 zijn de drie algemene formaten te zien die de instructies kunnen hebben. Voor alle voorbeelden is hetzelfde formaat gebruikt om een hexadecimaal getal voor te stellen: 0xhhh (waarin "h" een hexadecimale digit voorstelt).

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0 (store result in W) d = 1 (store result in file register 'f') Default is d = 1
label	Label name
TOS	Top of Stack
PC	Program Counter
WDT	Watchdog Timer Counter
$\overline{TO}$	Time-Out bit
$\overline{PD}$	Power-Down bit
dest	Destination, either the W register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

Tabel 7/6.5.5-14: Opcode field-beschrijvingen.

Opmerking
Voor de uitgewerkte instructieset (33 mogelijkheden) wordt verwezen naar de eerder behandelde PIC16C5x-familie (de figuren 7/6.5.2-35a tot en met -35i).

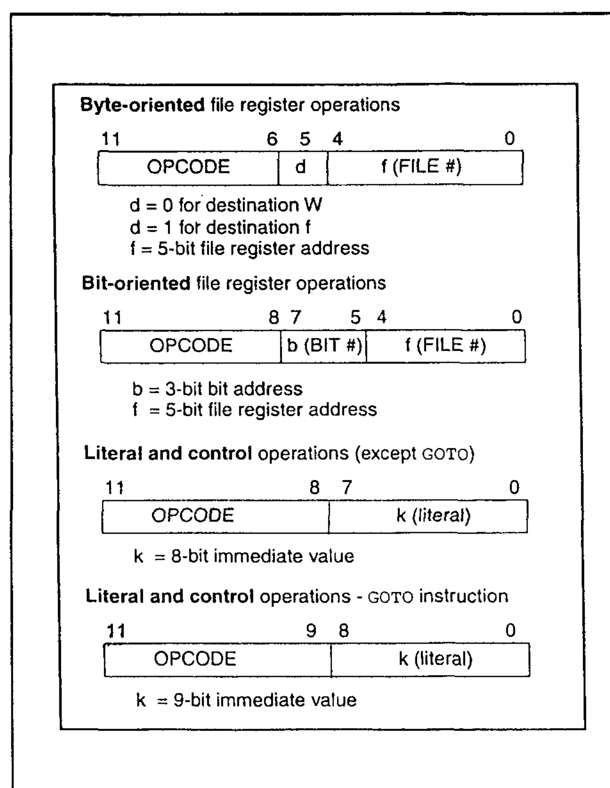
Ontwikkelings-gereedschappen

Inleiding
Voor de PIC16/17 microcontrollers staat een groot aantal ontwikkelings-tools ter beschikking.
Voor de PIC12C5xx-familie zijn dat er wat minder:

- PICMASTER:
  (EM167015/EM167101) een Universele In-Circuit Emulator, geschikt voor de PIC12C5xx, PIC14C000, PIC16C5x, PIC16Cxxx en PIC17Cxx families.
- PROMATE:
  (DV007003) Universele Programmer voor de PIC16C5x, PIC16Cxxx, PIC17Cxx, PIC14C000 en PIC12C5xx families.
- PICSTART:
  (DV003001) Entry Level Development System (via de RS232-poort) voor de PIC12C5xx, PIC14C000, PIC16C5x, PIC16Cxxx en PIC17Cxx families tot maximaal 40 pennen.
- MPLAB:
  (SW007002) Integrated Development Environment Software.
- MPLAB:
  (SW006005) (C Compiler).
- MPASM:
  Universele Macro Assembler voor alle PIC microcontrollers, inclusief de PIC12C5xx.
- Tenslotte is er natuurlijk de zogenaamde On-Line Support via Internet. De homepage van Microchip is: www.microchip.com. Het binnenhalen van files kan het beste gebeuren met behulp van het FTP-programma ftp.mchip.com/biz/mchip.

Elektrische en timing-eigenschappen

In de figuren 7/6.5.5-40 tot en met -44 en de tabellen 7/6.5.5-16 tot en met -24 wordt ten slotte een overzicht gegeven van de elektrische en timing-eigenschappen van de PIC12C5xx-familie.



Tabel 7/6.5.5-15: Algemene formaten van de instructies.

6.5 PIC-typen 8 bit microcontrollers

Ambient Temperature under bias	–40°C to +125°C
Storage Temperature	–65°C to +150°C
Voltage on V_{DD} with respect to V_{SS}	0 to +7.5 V
Voltage on $\overline{MCLR}$ with respect to V_{SS}	0 to +14 V
Voltage on all other pins with respect to V_{SS}	–0.6 V to (V_{DD} + 0.6 V)
Total Power Dissipation[1]	700 mW
Max. Current out of V_{SS} pin	200 mA
Max. Current into V_{DD} pin	150 mA
Input Clamp Current, I_{IK} ($V_I < 0$ or $V_I > V_{DD}$)	±20 mA
Output Clamp Current, I_{OK} ($V_O < 0$ or $V_O > V_{DD}$)	±20 mA
Max. Output Current sunk by any I/O pin	25 mA
Max. Output Current sourced by any I/O pin	25 mA
Max. Output Current sourced by I/O port (GPIO)	100 mA
Max. Output Current sunk by I/O port (GPIO)	100 mA

Note 1: Power Dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

Tabel 7/6.5.5-16: Maximaal toegelaten waarden voor de PIC12C5xx.

6.5 PIC-typen 8 bit microcontrollers

DC Characteristics
Power Supply Pins

Standard Operating Conditions (unless otherwise specified)
Operating Temperature 0°C ≤ TA ≤ +70°C (commercial)
–40°C ≤ TA ≤ +85°C (industrial)
–40°C ≤ TA ≤ +125°C (extended)

Characteristic	Sym	Min	Typ(1)	Max	Units	Conditions
Supply Voltage	VDD	2.5		5.5	V	FOSC = DC to 4 MHz (Commercial/ Industrial)
		3.0		5.5	V	FOSC = DC to 4 MHz (Extended)
RAM Data Retention Voltage(2)	VDR		1.5*		V	Device in SLEEP mode
VDD Start Voltage to ensure Power-on Reset	VPOR		VSS		V	See section on Power-on Reset for details
VDD Rise Rate to ensure Power-on Reset	SVDD	0.05*			V/ms	See section on Power-on Reset for details
Supply Current(3)	IDD	—	1.8	2.4	mA	XT and EXTRC options (Note 4) FOSC = 4 MHz, VDD = 5.5V
		—	1.8	2.4	mA	INTRC Option FOSC = 4 MHz, VDD = 5.5V
		—	15	27	µA	LP OPTION, Commercial Temperature FOSC = 32 kHz, VDD = 3.0V, WDT disabled
		—	19	35	µA	LP OPTION, Industrial Temperature FOSC = 32 kHz, VDD = 3.0V, WDT disabled
		—	19	35	µA	LP OPTION, Extended Temperature FOSC = 32 kHz, VDD = 3.0V, WDT disabled
Power-Down Current (5)	IPD					
WDT Enabled		—	4	12	µA	VDD = 3.0V, Commercial
		—	4	14	µA	VDD = 3.0V, Industrial
		—	5	22	µA	VDD = 3.0V, Extended
WDT Disabled		—	0.25	4	µA	VDD = 3.0V, Commercial
		—	0.25	5	µA	VDD = 3.0V, Industrial
		—	2	18	µA	VDD = 3.0V, Extended

* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is based on characterization results at 25°C. This data is for design guidance only and is not tested.

2: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

3: The supply current is mainly a function of the operating voltage and frequency. Other factors such as bus loading, oscillator type, bus rate, internal code execution pattern, and temperature also have an impact on the current consumption.

a) The test conditions for all IDD measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tristated, pulled to Vss, TOCKI = VDD, $\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

b) For standby current measurements, the conditions are the same, except that the device is in SLEEP mode.

4: Does not include current through Rext. The current through the resistor can be estimated by the formula: IR = VDD/2Rext (mA) with Rext in kOhm.

5: The power down current in SLEEP mode does not depend on the oscillator type. Power down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

Tabel 7/6.5.5-17: Gelijkspanningskenmerken van de PIC12C5xx (alleen de voedings-aansluitingen).

6.5 PIC-typen 8 bit microcontrollers

DC Characteristics All Pins Except Power Supply Pins	Standard Operating Conditions (unless otherwise specified) Operating Temperature 0°C ≤ T_A ≤ +70°C (commercial) –40°C ≤ T_A ≤ +85°C (industrial) –40°C ≤ T_A ≤ +125°C (extended) Operating Voltage V_{DD} range is described in Section 10.1.

Characteristic	Sym	Min	Typ[1]	Max	Units	Conditions
Input Low Voltage	V_{IL}					
I/O ports		Vss		0.8	V	Pin at hi-impedance 4.5V < V_{DD} ≤ 5.5V
		Vss		0.15 V_{DD}	V	Pin at hi-impedance 3.0V < V_{DD} ≤ 4.5V
$\overline{MCLR}$ and GP2 (Schmitt Trigger)		Vss		0.15 V_{DD}	V	
OSC1		Vss		0.15 V_{DD}	V	EXTRC option only[4]
OSC1		Vss		0.3 V_{DD}	V	XT and LP options
Input High Voltage	V_{IH}					
I/O ports		0.25V_{DD}+0.8V		V_{DD}	V	2.5V < V_{DD} ≤ 4.5V
		2.0		V_{DD}	V	4.5V < V_{DD} ≤ 5.5V[5]
		0.2V_{DD}+1V		V_{DD}	V	Full V_{DD} range[5]
$\overline{MCLR}$ and GP2 (Schmitt Trigger)		0.85 V_{DD}		V_{DD}	V	
OSC1 (Schmitt Trigger)		0.85 V_{DD}		V_{DD}	V	EXTRC option only[4]
		0.7 V_{DD}		V_{DD}	V	XT and LP options
IPUR						
Input Leakage Current[2,3]	I_{IL}					For V_{DD} ≤ 5.5V
I/O ports		–1	0.5	+1	µA	Vss ≤ V_{PIN} ≤ V_{DD}, Pin at hi-impedance
$\overline{MCLR}$		20	130	250	µA	V_{PIN} = Vss + 0.25V[2]
			0.5	+5	µA	V_{PIN} = V_{DD}
OSC1		–3	0.5	+3	µA	Vss ≤ V_{PIN} ≤ V_{DD}, XT and LP options
Output Low Voltage	Vol					
I/O ports				0.6	V	I_{OL} = 8.7 mA, V_{DD} = 4.5V
Output High Voltage[3,4]	VoH					
I/O ports		V_{DD} –0.7			V	I_{OH} = –5.4 mA, V_{DD} = 4.5V

* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is based on characterization results at 25°C. This data is for design guidance only and is not tested.

2: The leakage current on the $\overline{MCLR}$/VPP pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltage.

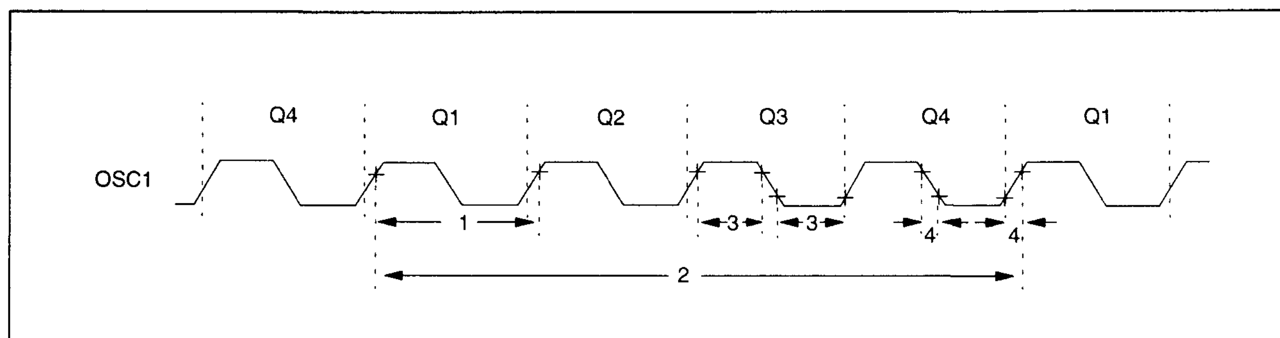
3: Negative current is defined as coming out of the pin.

4: For PIC12C5XX devices, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC12C5XX be driven with external clock in RC mode.

5: The user may use the better of the two specifications.

Tabel 7/6.5.5-18: Gelijkspanningskenmerken van de PIC12C5xx (de overige aansluitpennen).

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-40: Externe clock-timing (zie tabel 7/6.5.5-19).

Parameter No.	Sym	Characteristic	Min	Typ(1)	Max	Units	Conditions
	Fosc	External CLKIN Frequency(2)	DC	—	4	MHz	EXTRC osc mode
			DC	—	4	MHz	XT osc mode
			DC	—	200	kHz	LP osc mode
		Oscillator Frequency(2)	DC	—	4	MHz	EXTRC osc mode
			0.1	—	4	MHz	XT osc mode
			DC	—	200	kHz	LP osc mode
1	Tosc	External CLKIN Period(2)	250	—	—	ns	EXTRC osc mode
			250	—	—	ns	XT osc mode
			5	—	—	ms	LP osc mode
		Oscillator Period(2)	250	—	—	ns	EXTRC osc mode
			250	—	10,000	ns	XT osc mode
			5	—	—	ms	LP osc mode
2	Tcy	Instruction Cycle Time(3)	—	4/Fosc	—	—	
3	TosL, TosH	Clock in (OSC1) Low or High Time	50*	—	—	ns	XT oscillator
			2*	—	—	ms	LP oscillator
4	TosR, TosF	Clock in (OSC1) Rise or Fall Time	—	—	25*	ns	XT oscillator
			—	—	50*	ns	LP oscillator

* These parameters are characterized but not tested.

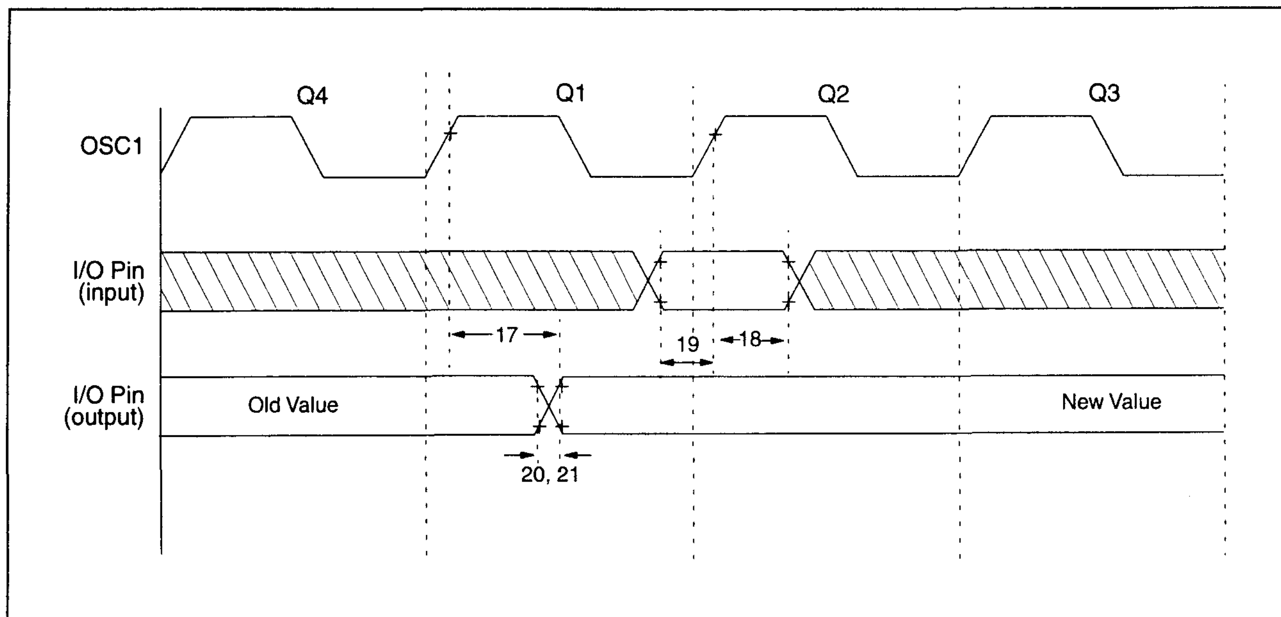
Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

2: All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption.
When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

3: Instruction cycle period (Tcy) equals four times the input oscillator time base period.

Tabel 7/6.5.5-19: Eisen die aan de externe clock worden gesteld (zie figuur 7/6.5.5-40).

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-41: I/O-timing bij de PIC12C5xx (zie tabel 7/6.5.5-20).

Parameter No.	Sym	Characteristic	Min	Typ(1)	Max	Units
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid(3)	—	—	100*	ns
18	TosH2iol	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	TBD	—	—	ns
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)	TBD	—	—	ns
20	TioR	Port output rise time(3)	—	10	25**	ns
21	TioF	Port output fall time(3)	—	10	25**	ns

* These parameters are characterized but not tested.

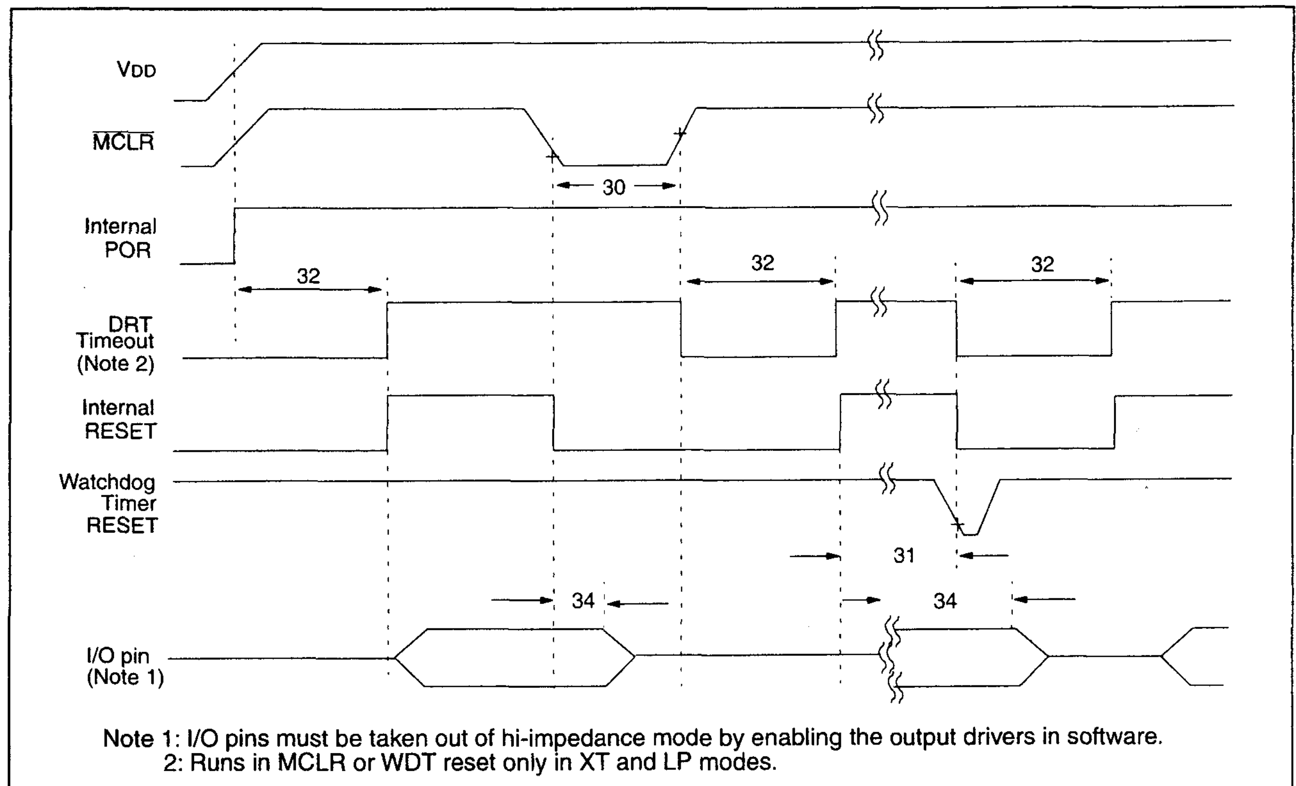
** These parameters are design targets and are not tested. No characterization data available at this time.

Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

2: Measurements are taken in EXTRC mode.

Tabel 7/6.5.5-20: Eisen die aan de I/O-timing worden gesteld (zie ook figuur 7/6.5.5-41).

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.5-42: Reset, Watchdog Timer en Device Reset Timer timing (zie tabel 7/6.5.5-21).

Parameter No.	Sym	Characteristic	Min	Typ(1)	Max	Units	Conditions
30	TmcL	$\overline{\text{MCLR}}$ Pulse Width (low)	2000*	—	—	ns	VDD = 5 V
31	Twdt	Watchdog Timer Time-out Period (No Prescaler)	9*	18*	30*	ms	VDD = 5 V (Commercial)
32	TDRT	Device Reset Timer Period(2)	9*	18*	30*	ms	VDD = 5 V (Commercial)
34	Tioz	I/O Hi-impedance from $\overline{\text{MCLR}}$ Low	—	—	2000*	ns	

* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Tabel 7/6.5.5-21: De timing van Reset, Watchdog Timer en Device Reset Timer (zie figuur 7/6.5.5-42).

6.5 PIC-typen 8 bit microcontrollers

Oscillator Configuration	POR Reset	Subsequent Resets
IntRC & ExtRC	18 ms (typical)	300 μs (typical)
XT & LP	18 ms (typical)	18 ms (typical)

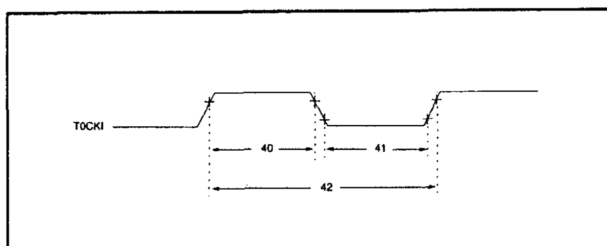
Tabel 7/6.5.5-22: DRT (Device Reset Timer) Periode.

Parameter No.	Sym	Characteristic	Min	Typ(1)	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width - No Prescaler	0.5 T_{CY} + 20*	—	—	ns	
		- With Prescaler	10*	—	—	ns	
41	Tt0L	T0CKI Low Pulse Width - No Prescaler	0.5 T_{CY} + 20*	—	—	ns	
		- With Prescaler	10*	—	—	ns	
42	Tt0P	T0CKI Period	20 or $\frac{T_{CY} + 40}{N}$*	—	—	ns	Whichever is greater. N = Prescale Value (1, 2, 4,..., 256)

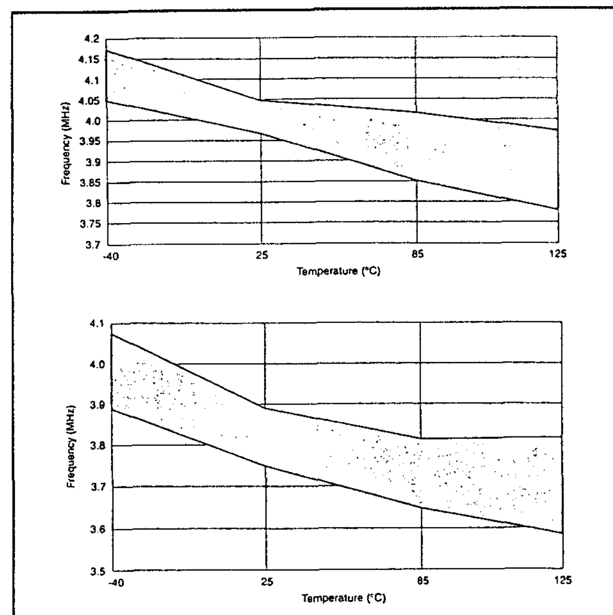
* These parameters are characterized but not tested.

Note 1: Data in the Typical ("Typ") column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Tabel 7/6.5.5-23: Eisen die aan de TIMER0-clock worden gesteld (zie figuur 7/6.5.5-43).



Figuur 7/6.5.5-43: Timer0 clock-timing (zie tabel 7/6.5.5-23).



Figuur 7/6.5.5-44: Gecalibreerde interne RC-oscillator frequentie als functie van de temperatuur (a: V_{DD} = 5,0 V, b: V_{DD} = 3,0 V).

6.5 PIC-typen 8 bit microcontrollers

VDD (Volts)	Temperature (°C)	Min	Typ	Max	Units
GP0/GP1					
2.5	-40	50K	62K	84K	Ω
	25	63K	73K	84K	Ω
	85	63K	80K	84K	Ω
	125	63K	81K	84K	Ω
5.5	-40	19K	22K	27K	Ω
	25	25K	27K	31K	Ω
	85	28K	33K	40K	Ω
	125	31K	36K	43K	Ω
GP3					
2.5	-40	490K	710K	965K	Ω
	25	610K	890K	1.2M	Ω
	85	769K	1.1M	1.5M	Ω
	125	810K	1.2M	1.6M	Ω
5.5	-40	310K	410K	510K	Ω
	25	360K	470K	580K	Ω
	85	430K	550K	670K	Ω
	125	465K	585K	715K	Ω

* These parameters are characterized but not tested.

Tabel 7/6.5.5-24: **Selectietabel van bruikbare optrekweerstanden.**

6.5 PIC-typen 8 bit microcontrollers

7/6.5.6

Type-beschrijving van de PIC12CE5xx-typen

Algemene gegevens

Inleiding

De PIC12CE5xx is, net als de PIC12C5xx, een 8 bit, statische, op EPROM/ROM gebaseerde CMOS microcontroller met slechts 8 aansluitpennen. De PIC12CE5xx-serie is volkomen identiek aan de PIC12C5xx-familie, maar heeft een extra EEPROM aan boord. Voor de algemene beschrijving van deze microcontroller wordt dan ook verwezen naar de PIC12C5xx.

De PIC12CE5xx microcontrollers zijn verkrijgbaar in een UV-wisbare CERDIP-behuizing (voor het ontwikkelen van code) en in de goedkopere OTP-behuizingen (PDIP of SOIC) voor de productiedoeleinden.

Ook de PIC12CE5xx-typen worden ondersteund door een macro-assembler, software simulator, in-circuit emulator, C-compiler, fuzzy-logic tools, een goedkope ontwikkelprogrammer en een complete programmer. Alle tools zijn bruikbaar op IBM PC's en vergelijkbare typen.

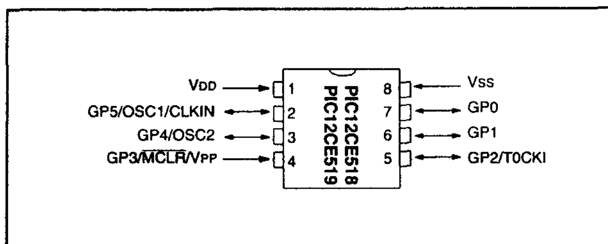
Algemene gegevens

- behandelde typen: PIC12CE518 en PIC12CE519
- 33 single-word instructies
- één cyclus per instructie (1 μs) (branches 2 cycli)
- snelheid: DC - 4 MHz clock input (minimale instructie-cyclus: 1 μs)
- 12 bits instructies, 8 bits data
- 7 special function hardware registers
- 2-niveaus diepe hardware stack
- directe, indirecte en relatieve adresseringsmodes voor data en instructies
- max. 1 miljoen x wissen/schrijven in EEPROM data-geheugen
- min. 40 jaar data vasthouden in EEPROM
- interne 4 MHz RC-oscillator met programmeerbare calibratie
- in-circuit seriële programmering via 2 pennen
- 8 bit real-time clock/counter (TMR0) met 8 bit programmeerbare prescaler
- power-on reset (POR) en device reset timer (DRT)
- watchdog timer (WDT) met eigen on-chip RC-oscillator
- programmeerbare code-beveiliging
- energiebesparende SLEEP-mode
- ontwaken uit SLEEP-mode bij signaalverandering op pen
- interne “zwakke” optrekking van I/O-pennen
- interne pull-up van $\overline{MCLR}$-pen
- selecteerbare oscillator-opties:
  - INTRC: interne 4 MHz RC-oscillator
  - EXTRC: externe RC-oscillator
  - XT: standaard kristal/resonator
  - LP: energie besparend laagfrequent kristal
- CMOS: volledig statisch ontwerp
- voedingsspanning: commercieel en industrieel: 3,0 V - 5,5 V uitgebreid: 4,5 V - 5,5 V
- laag energie-verbruik: <2 mA typ. (5 V, 4 MHz) 15 μA typ. (3V, 32 kHz) <1 μA typ. (standby)
- fabrikant: Microchip Technology Inc.

6.5 PIC-typen 8 bit microcontrollers

Device	Memory		
	EPROM Program	RAM Data	EEPROM Data
PIC12CE518	512 x 12	25 x 8	16 x 8
PIC12CE519	1024 x 12	41 x 8	16 x 8

Figuur 7/6.5.6-1: Verkorte kenmerken van de PIC12CE5xx-typen.



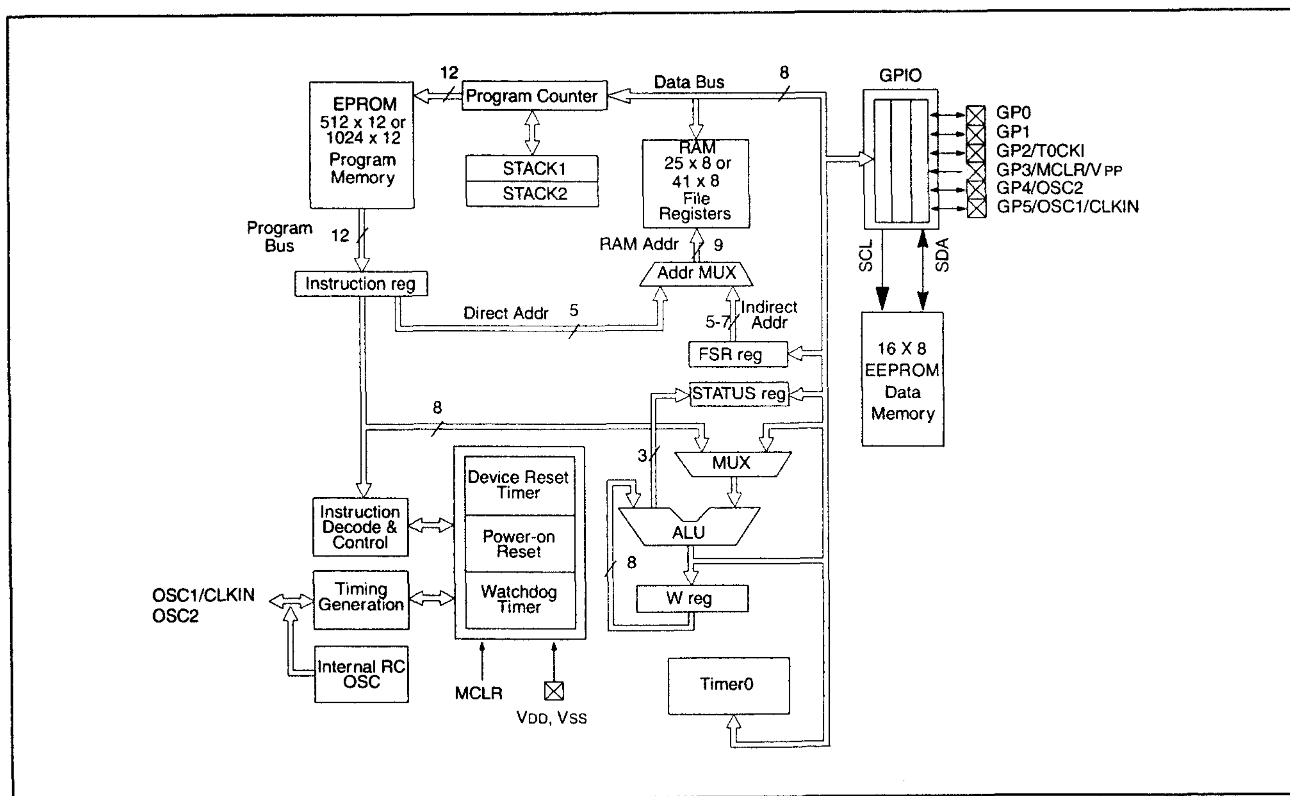
Figuur 7/6.5.6-2: Aansluitingen van de PIC12CE518 en PIC12CE519 (8-pens PDIP, SOIC en CER-DIP uitvoering (met venster).

Toepassingen

De PIC12CE5xx-typen zijn onder andere zeer geschikt voor sensor-systemen, gasdetectoren, beveiligingssystemen en draagbare zenders/ontvangers. De EPROM-technologie maakt zeer snelle aanpassing van de programma's (zender-codes, instellingen, ontvangstfrequenties en dergelijke) mogelijk.

Architectuur

Ook bij de PIC12CE5xx-familie wordt een Harvard architectuur gebruikt, waarbij de toegang tot programma en data via aparte bussen plaats vindt. Hierdoor wordt niet alleen de bandbreedte verbeterd ten opzichte van de traditionele von Neumann architectuur, maar kunnen ook 12 bit instructies en 8 bit datawoorden worden gebruikt. Met de PIC12CE518 kan 512 x 12 programmageheugen worden geadresseerd en met de PIC12CE519 1 k x 12.



Figuur 7/6.5.6-3: Vereenvoudigd blokschema van de PIC12CE5xx microcontrollers (zie extra EEPROM).

Het programma-geheugen is intern. De register-files en het data-geheugen kunnen direct of indirect worden geadresseerd. Alle registers voor speciale functies (inclusief de program-counter) bevinden zich in het data-geheugen.
De PIC12CE5xx is volkomen identiek aan de PIC12C5xx, maar bevat daarnaast nog een 16 x 8 EEPROM geheugen-array voor de opslag van niet-vluchtige informatie, zoals calibratie-gegevens of beveiligingscodes. Dit geheugen kan 1 miljoen keer worden gewist en ge(her)programmeerd, terwijl de data gedurende minimaal 40 jaar wordt vastgehouden.
De PIC12CE5xx bevat een 8 bit ALU en werk-register. De ALU is een algemene rekenkundige eenheid voor het uitvoeren van rekenkundige en Boole'se bewerkingen tussen data in het werkregister en data in een willekeurige register-file. De ALU kan optellen, aftrekken, schuiven en logische operaties uitvoeren. Tenzij anders vermeld zijn rekenkundige operaties two's complement van aard. Bij 2-operand instructies is gewoonlijk één operand het werkregister (W).

Name	DIP Pin #	SOIC Pin #	I/O/P Type	Buffer Type	Description
GP0	7	7	I/O	TTL/ST	Bi-directional I/O port/ serial programming data. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. This buffer is a Schmitt Trigger input when used in serial programming mode.
GP1	6	6	I/O	TTL/ST	Bi-directional I/O port/ serial programming clock. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. This buffer is a Schmitt Trigger input when used in serial programming mode.
GP2/T0CKI	5	5	I/O	ST	Bi-directional I/O port. Can be configured as T0CKI.
GP3/$\overline{MCLR}$/VPP	4	4	I	TTL	Input port/master clear (reset) input/programming voltage input. When configured as $\overline{MCLR}$, this pin is an active low reset to the device. Voltage on $\overline{MCLR}$/VPP must not exceed VDD during normal device operation. Can be software programmed for internal weak pull-up and wake-up from SLEEP on pin change. Weak pull-up always on if configured as $\overline{MCLR}$
GP4/OSC2	3	3	I/O	TTL	Bi-directional I/O port/oscillator crystal output. Connections to crystal or resonator in crystal oscillator mode (XT and LP modes only, GPIO in other modes).
GP5/OSC1/CLKIN	2	2	I/O	TTL/ST	Bidirectional IO port/oscillator crystal input/external clock source input (GPIO in Internal RC mode only, OSC1 in all other oscillator modes). TTL input when GPIO, ST input in external RC oscillator mode.
VDD	1	1	P	—	Positive supply for logic and I/O pins
Vss	8	8	P	—	Ground reference for logic and I/O pins

Legend: I = input, O = output, I/O = input/output, P = power, — = not used, TTL = TTL input,
ST = Schmitt Trigger input

Tabel 7/6.5.6-1: Aansluitingen en signaalfuncties van de PIC12CE518 en PIC12CE519.

De andere is óf een file-register óf een immediate constante. Bij 1-operand instructies is de operand het W-register of een file-register.
Het W-register is een 8 bit werkregister dat voor ALU-operaties wordt gebruikt. Het is niet adresseerbaar.
Afhankelijk van de uitgevoerde instructie kan de ALU de waarden van de Carry (C), Digit Carry (DC) en Zero (Z) bits in het statusregister beïnvloeden. De C en DC bits werken bij aftrekken respectievelijk als borrow en digit borrow out-bit, zie bijvoorbeeld de SUBWF en ADDWF instructies.

Overeenkomstige kenmerken met de PIC12C5xx-familie
Voor een groot aantal van de volgende kenmerken wordt verwezen naar de gelijknamige beschrijvingen bij de PIC12C5xx-serie, er wordt dan ook verwezen naar figuren en tabellen uit het hoofdstuk 7/6.5.5.

Clock-schema/Instructie-cyclus
Het intern door vier gedeelde clock-ingangssignaal (OSC1/CLKIN) genereert 4 niet-overlappende kwadratuur clock-signalen (Q1, Q2, Q3 en Q4), zie de figuren 7/6.5.5-5 en 7/6.5.5-6.

Instructie-afhandeling/pijplijnen
Een instructie-cyclus bestaat uit vier Q-cycli (Q1, Q2, Q3 en Q4).
Door het pijplijnen is voor elke instructie slechts één cyclus nodig.

Geheugen-organisatie en registers

Inleiding
Het geheugen van de PIC12CE5xx is georganiseerd in programma-geheugen en data-geheugen. Bij meer dan 512 bytes programma-geheugen (hier dus de PIC12CE519) wordt "paging" toegepast. Toegang tot deze pagina's wordt verkregen met behulp van één bit in het STATUS-register. Data-geheugenbanken worden bereikt met behulp van het File Select Register (FSR).

Programma-geheugen
De PIC12CE5xx heeft een 12 bit Program Counter die in staat is om 2 k geheugenwoorden van elk 12 bit te adresseren.
Bij de PIC12CE518 zijn alleen de eerste 512 x 12 (0000h - 01FFh) en bij de PIC12CE519 alleen de eerste 1 k x 12 (0000h - 03FFh) plaatsen fysiek geïmplementeerd (zie figuur 7/6.5.5-7).

Data-geheugen
Het data-geheugen is opgebouwd uit RAM-registers.
Daarom wordt het data-geheugen gespecificeerd door zijn register-file. De register-file is verdeeld in twee functionele groepen: speciale functie-registers en algemene (general purpose) registers. Bij de PIC12CE518 bestaat de register-file uit 7 speciale functie-registers en 25 algemene registers (figuur 7/6.5.5-8).
De PIC12CE519 heeft bovendien nog 16 algemene registers die door middel van banking geadresseerd kunnen worden (figuur 7/6.5.5-9).

General Purpose Register File
De algemene register-file kan direct of indirect worden geadresseerd via het file-select register FSR.

De Special Function Registers
De Special Function Registers worden door de CPU en perifere functies gebruikt om de werking van de microcontroller te regelen (zie tabel 7/6.5.6-2).

Het Status Register
Het Status Register bevat de rekenkundige toestand van de ALU, de RESET status en het pagina-preselect-bit voor programma-geheugens, groter dan 512 woorden (zie figuur 7/6.5.5-10).

Het Option Register

Het Option Register is een 8 bit write-only register dat allerlei besturingsbits bevat om de TMR0/WDT prescaler en Timer0 te configureren (figuur 7/6.5.5-11).

OSCCAL Register

Het Oscillator Calibratie Register (OSCCAL) wordt gebruikt om de interne 4 MHz oscillator te calibreren. Dit register bevat vier bits voor "fijn" calibratie en twee bits voor verhogen of verlagen van de frequentie (zie figuur 7/6.5.6-4).

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on $\overline{MCLR}$ and WDT Reset	Value on Wake-up on Pin Change
N/A	TRIS	—	—		I/O control registers					--11 1111	--11 1111	--11 1111
N/A	OPTION	Contains control bits to confi gure Timer0, Timer0/WDT prescaler, wake-up on change , and weak pull-ups								1111 1111	1111 1111	1111 1111
00h	INDF	Uses contents of FSR to address data memor y (not a physical register)								xxxx xxxx	uuuu uuuu	uuuu uuuu
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu	uuuu uuuu
02h(1)	PCL	Low order 8 bits of PC								1111 1111	1111 1111	1111 1111
03h	STATUS	GPWUF	—	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu	100q quuu
04h	FSR (12CE518)	Indirect data memor y address pointer								111x xxxx	111u uuuu	111u uuuu
04h	FSR (12CE519)	Indirect data memor y address pointer								110x xxxx	11uu uuuu	11uu uuuu
05h	OSCCAL (12CE518/ 12CE519)	CAL7	CAL6	CAL5	CAL4	CALFST	CALSLW	—	—	0111 00--	uuuu uu--	uuuu uu--
06h	GPIO	SCL	SDA	GP5	GP4	GP3	GP2	GP1	GP0	11xx xxxx	11uu uuuu	11uu uuuu

Legend: Shaded bo xes = unimplemented or un used, – = unimplemented, read as '0' (if applicab le)
x = unknown, u = unchanged, q = see the tab les in Section 8.7 for possible v alues.

Note 1: The upper b yte of the Prog ram Counter is not directly accessib le. See Section 4.6 for an explanation of ho w to access these bits .

Tabel 7/6.5.6-2: Overzicht van de Special Function Registers (SFR) van de PIC12CE5xx-serie.

R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	U-0	U-0
CAL3	CAL2	CAL1	CAL0	CALFST	CALSLW	—	—
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7-4: **CAL<3:0>:** Fine calibration

bit 3: **CALFST:** Calibration Fast
1 = Increase frequency
0 = No change

bit 2: **CALSLW:** Calibration Slow
1 = Decrease frequency
0 = No change

bit 1-0: **Unimplemented:** Read as '0'

Note: If CALFST = 1 and CALSLW = 1, CALFST has precedence

Figuur 7/6.5.6-4: Inrichting van het OSCCAL-register (adres 8Fh).

6.5 PIC-typen 8 bit microcontrollers

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on $\overline{MCLR}$ and WDT Reset	Value on Wake-up on Pin Change
N/A	TRIS	—	—	I/O control registers						--11 1111	--11 1111	--11 1111
N/A	OPTION	$\overline{GPWU}$	$\overline{GPPU}$	TOCS	TOSE	PSA	PS2	PS1	PS0	1111 1111	1111 1111	1111 1111
03H	STATUS	GPWUF	—	PA0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	000q quuu	100q quuu
06h	GPIO	SCL	SDA	GP5	GP4	GP3	GP2	GP1	GP0	11xx xxxx	11uu uuuu	11uu uuuu

Legend: Shaded cells not used b y Port Registers, read as '0', — = unimplemented, read as '0', x = unknown, u = unchanged,

Tabel 7/6.5.6-3: Overzicht van de poort-registers van de PIC12CE5xx.

Program Counter
Bij het uitvoeren van instructie krijgt de program counter (PC) het adres van de volgende instructie (zie figuur 7/6.5.5-12).

De effecten van RESET
Een RESET zet Program Counter, waardoor de laatste lokatie op de laatste pagina wordt geadresseerd (:de oscillator calibratie instructie).

Stack
De PIC12CE5xx heeft een 12 bit brede hardware push/pop stack.

Indirecte data adressering, INDF en FSR Registers
Het INDF register is geen fysiek register. Adresseren van INDF betekent dus het adresseren van het register waarvan het adres in het FSR-register staat (FSR is een pointer). Dit is indirect adresseren (zie figuur 7/6.5.5-13).

I/O-poort
Net als andere registers kan het I/O-register worden uitgelezen en beschreven onder programma-besturing. Na een RESET zijn alle I/O-poorten gedefinieerd als ingang (hoog-impedant).

GPIO
GPIO is een 8 bit I/O-register waarvan alleen de laagste 6 bits worden gebruikt (GP5:GP0).

TRIS-register
Het output-driver besturingsregister wordt geladen met de inhoud van het W-register bij de uitvoering van de TRIS f instructie.
Opmerking:
Door een lees-operatie van de poorten worden de pennen en NIET de uitgangs-datalatches uitgelezen.

Overige eigenschappen

I/O Interfacing
Alle I/O-poorten (behalve GP3) kunnen voor ingangs- en uitgangs-operaties worden gebruikt (figuur 7/6.5.5-15).

Bi-directionele I/O-poorten
Sommige instructies werken intern als lezen, gevolgd door schrijf-operaties.
De BCF en BSF instructies lezen bijvoorbeeld de gehele poort in de CPU, voeren de bit-operatie uit en zetten het resultaat op de uitgang.
In voorbeeld 3 (figuur 7/6.5.5-16) is het effect van twee opvolgende read-modify-write instructies op een I/O-poort te zien.

Opeenvolgende operaties op I/O-poorten
Het feitelijke schrijven naar een I/O-poort vindt plaats aan het einde van een instructiecyclus. Bij het lezen moet de data al direct aanwezig zijn (figuur 7/6.5.5-17 en tabel 7/6.5.5-4).

TIMER0 MODULE en TMR0 REGISTER

De Timer0-module (figuur 7/6.5.5-18) heeft de volgende eigenschappen:
- 8 bit timer/counter register, TMR0 - lees- en schrijfbaar
- 8 bit met software programmeerbare prescaler
- Interne of externe clock-select - edge select voor externe clock

Als in het TMR0-register wordt geschreven, wordt het incrementeren gedurende de volgende twee cycli gesperd (figuren 7/6.5.5-19 en -20). De counter-mode wordt geselecteerd door het T0CS-bit (OPTION) te setten. De prescaler kan worden gebruikt door de Timer0-module of de Watchdog Timer, maar niet door beide tegelijk.

Gebruik van Timer0 met een externe clock

Wanneer voor Timer0 een extern clocksignaal wordt gebruikt, worden daaraan eisen gesteld om synchronisatie op de interne phase-clock (T_{OSC}) mogelijk te maken.

Externe clock synchronisatie

Als geen prescaler wordt gebruikt is het externe clock-signaal gelijk aan het prescaler uitgangssignaal. Voor synchronisatie zie figuur 7/6.5.5-21.

Timer0 increment vertraging

Aangezien de uitgang van de prescaler op de interne clock's wordt gesynchroniseerd, is er een vertraging (figuur 7/6.5.5-21).

Option-register effect op GP2 TRIS

Als Timer0 wordt uitgelezen wordt deze poort een uitgang.

Prescaler

Er is een 8 bit teller (figuur 7/6.5.5-22) die dient als prescaler voor de Timer0-module of als postscaler voor de Watchdog Timer (WDT).
Deze kan worden toegewezen aan de Timer0-module of aan de Watchdog Timer, maar niet aan beide tegelijk.

Veranderen van prescaler-toewijzing

De toewijzing van de prescaler gebeurt onder software-besturing, zie ook figuur 7/6.5.5-23 en figuur 7/6.5.5-24.

Speciale eigenschappen van de CPU

De PIC12CE5xx-familie heeft verschillende speciale voorzieningen voor real-time toepassingen. De Watchdog Timer loopt op zijn eigen RC-oscillator.
Bij gebruik van de XT of LP oscillator-opties levert de DRT een vertraging van 18 ms om de chip gereset te houden totdat de kristaloscillator stabiel is. Om uit de SLEEP mode te ontwaken is een verandering op een ingangspen of een Watchdog Timer time-out voldoende.

Configuratiebits

Het configuratiewoord van de PIC12CE5xx bestaat uit 5 bits, zie figuur 7/6.5.5-25.

Oscillator-configuraties

De PIC12CE5xx kan in vier verschillende oscillator-modes werken:
- LP: Low Power Crystal
- XT: Crystal/Resonator
- INTRC: Interne 4 MHz Oscillator
- EXTRC: Externe Resistor/Capacitor.

Crystal Oscillator/Ceramic Resonator: in de XT of LP modes is een kristal of een ceramische resonator verbonden met de GP5/OSC1/CLKIN en GP4/OSC2 pennen (zie figuur 7/6.5.5-26).
In de tabellen 7/6.5.5-5 en -6 zijn de waarden voor de condensatoren te zien. In de XT of LP mode is ook externe clock-aansturing op GP5/OSC1/CLKIN mogelijk (figuur 7/6.5.5-27).
Externe kristal-oscillator: als externe kristal-oscillator kan een kant-en-klare oscillator of een eenvoudige, uit TTL-poorten opgebouwde oscillator worden gebruikt. Er is parallelle resonantie (figuur 7/6.5.5-28) en serie resonantie (figuur 7/6.5.5-29) mogelijk.
Externe RC-oscillator: voor tijdkritische toepassingen is de RC-optie (figuur 7/6.5.5-30) een goede keuze.

Interne RC-oscillator: de interne RC-oscillator levert een vaste (nominale) systeemclock van 4 MHz, die gecalibreerd kan worden.

Reset

De PIC12CE5xx kent verschillende soorten reset:

- Power-On Reset (POR);
- $\overline{MCLR}$ reset tijdens normaal bedrijf;
- $\overline{MCLR}$ reset tijdens SLEEP;
- WDT time-out reset bij normaal bedrijf;
- WDT time-out reset tijdens SLEEP;
- Wake-up uit SLEEP bij verandering op een pen.

Niet alle registers worden gereset.

In de tabellen 7/6.5.5-7 en -8 zijn de resettoestanden van alle registers, respectievelijk de speciale registers te zien en figuur 7/6.5.2-32 is het blokschema van de on-chip resetschakeling.

$\overline{MCLR}$ Enable geeft, indien niet geprogrammeerd, de externe $\overline{MCLR}$-functie vrij (figuur 7/6.5.5-31). Power-On Reset (POR): de PIC12CE5xx-familie heeft een Power-On Reset schakeling voor interne chip-resets (zie figuur 7/6.5.5-32). De Power-On Reset en de Device Reset Timer zijn nauw verwant. Zie power-up voorbeelden: figuur 7/6.5.5-33, -34 en -35. Device Reset Timer (DRT): telkens als de voedingsspanning terugkomt loopt de DRT (zie tabel 7/6.5.5-9).

Watchdog Timer (WDT)

WDT is een aparte, vrijlopende on-chip RC-oscillator zonder externe componenten (los van de externe RC-oscillator en de interne 4 MHz oscillator), zie tabel 7/6.5.2-10 en figuur 7/6.5.5-36. De WDT heeft een time-out periode van 18 ms (zonder prescaler). Met een prescaler kan een maximale time-out van 2,3 s worden gerealiseerd.

Time-Out volgorde,
Power Down en Wake-up uit SLEEP statusbits ($\overline{TO}$/PD/GPWUF)

De $\overline{TO}$, $\overline{PD}$ en GPWUF bits in het STATUS-register kunnen worden getest om te bepalen waardoor een RESET-conditie werd veroorzaakt (zie de tabellen 7/6.5.5-11 en -12).

Reset na Brown-Out

Bij een Brown-out komt de voedingsspanning (V_{dd}) tijdelijk onder de minimumwaarde. De schakeling dient dan gereset te worden met een externe brown-out beveiliging (zie de figuren 7/6.5.5-37 en -38).

Power-Down Mode (SLEEP)

De PIC12CE5xx kan tijdelijk op een laag vermogen worden gezet (SLEEP) en daar weer uit ontwaken door:

- een externe reset;
- een Watchdog Timer time-out;
- een verandering op een ingangspen.

Programma-verificatie / code-beveiliging

Bij een niet-geprogrammeerde codebeveiligingsbit kan het programma-geheugen worden uitgelezen.

ID-lokaties

Er zijn 4 geheugenplaatsen bestemd als ID-lokaties voor de opslag van checksum of andere code-identificatie-getallen.

In-circuit seriële programmering

De PIC12CE5xx zijn serieel programmeerbaar, ook als ze reeds zijn ingesoldeerd. Het programmeren gebeurt met twee lijnen voor clock en data plus drie lijnen voor voeding, aarde en de programmeerspanning. De schakeling komt in de program/verify-mode door de spanning op de $\overline{MCLR}$ (V_{pp})-pen te verhogen tot V_{IHH} en de GP1 en GP0 pennen LAAG te houden. Hierdoor wordt GP1 de programmeer-clock en GP0 de programmeer-data (zie figuur 7/6.5.5-39).

Werking van de EEPROM

EEPROM data-geheugen

De PIC12CE518 en PIC12CE519 hebben elk 16 bytes EEPROM aan boord. Deze EEPROM is toegankelijk via een 2-draads bus en transmissie-protocol (zie verderop).

De PIC12CE518 en PIC12CE519 zijn allebei voorzien van 16 bytes EEPROM data-geheugen (elektrisch wisbaar). Dit EEPROM data-geheugen wordt bediend via een bidirectionele 2-draads bus (zie figuur 7/6.5.6-3) en een data-transmissie protocol. Deze twee draden seriële data (SDA) en seriële clock (SCL) zijn toegewezen aan bit6 en bit7 van het GPIO-register (SFR 06h). In tegenstelling tot GP0 tot en met GP5 (die met de I/O-pennen zijn verbonden) gaan SDA en SCL alleen naar de interne EE-PROM. Voor de meeste toepassingen is het aanroepen van de volgende functies voldoende (zie figuur 7/6.5.6-5).

```
; Byte_Write: Byte write routine
;       Inputs: EEPROM Address      EEADDR
;               EEPROM Data         EEDATA
;       Outputs:    Return 01 in W if OK, else
;                   return 00 in W
;
; Read_Current: Read EEPROM at address
currently held by EE device.
;       Inputs: NONE
;       Outputs:    EEPROM Data     EEDATA
;                   Return 01 in W if OK, else
;                   return 00 in W
;
; Read_Random: Read EEPROM byte at supplied
address
;       Inputs: EEPROM Address      EEADDR
;       Outputs:    EEPROM Data     EEDATA
;                   Return 01 in W if OK,
;                   else return 00 in W
```

Figuur 7/6.5.6-5: Functies voor de bediening van het interne EEPROM data-geheugen.

Seriële data

SDA is een bidirectionele verbinding voor de overdracht van adressen en data. Voor normale data-overdracht mag SDA alleen veranderen als SCL LAAG is. Veranderingen bij een HOGE SCL zijn gereserveerd voor het aangeven van de START- en STOP-condities.

Seriële clock

De SCL-ingang wordt gebruikt om de data-overdracht van en naar de schakeling te synchroniseren.

Bus-karakteristieken

Bij het EEPROM data-geheugen wordt het volgende busprotocol gebruikt:

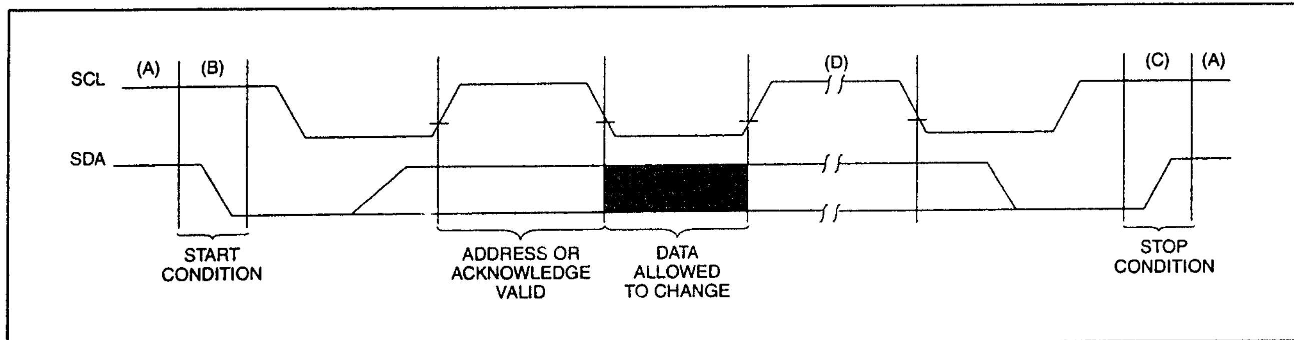
- Data mag alleen worden veranderd als de bus niet bezig is.
- Tijdens data-transport moet de datalijn stabiel blijven als de clock HOOG is. Veranderingen tijdens HOGE clock worden geïnterpreteerd als START of STOP.

Daarom zijn de volgende bus-condities gedefinieerd (figuur 7/6.5.6-6):

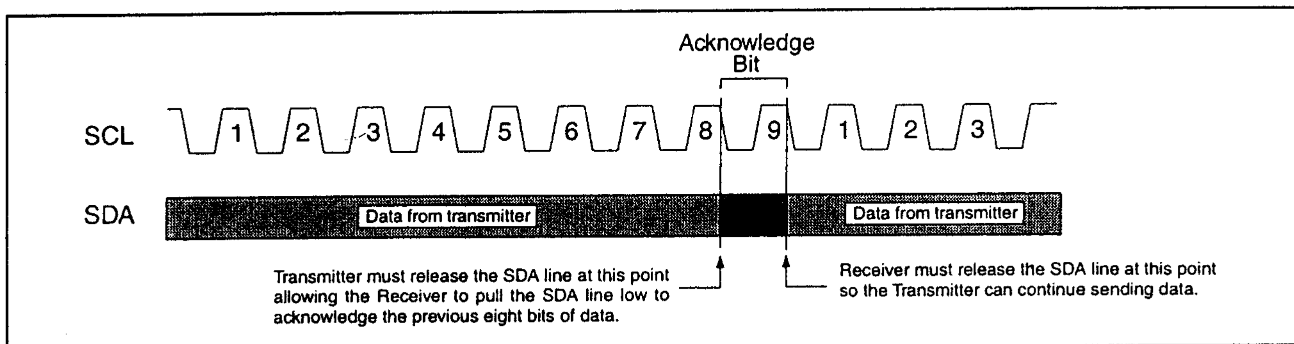
- BUS NOT BUSY (A):
  Zowel data als clock blijven HOOG.
- START DATA TRANSFER (B):
  Een HOOG-naar-LAAG overgang op de SDA-lijn, terwijl de clock (SCL) HOOG is, geeft een START-conditie aan. Alle commando's moeten hierdoor worden vooraf gegaan.
- STOP DATA TRANSFER (C):
  Een LAAG-naar-HOOG overgang op de SDA-lijn, terwijl de clock (SCL) HOOG is, komt overeen met een STOP-conditie. Alle operaties moeten worden beëindigd door een STOP-conditie.
- DATA VALID (D):
  De toestand op de datalijn vertegenwoordigt geldige data als de datalijn, na de START-conditie, stabiel is gedurende de tijd dat het clock-signaal HOOG is. De data op deze lijn mag veranderen als het clock-signaal LAAG is. Er is één bit data per clockpuls. Elke data-overdracht wordt geïnitialiseerd met een START-conditie en beëindigd met een STOP-conditie. Het aantal databytes dat tussen START en STOP wordt overgebracht, wordt bepaald door de master en is theoretisch onbegrensd.
- ACKNOWLEDGE:
  Elke (geadresseerde) ontvanger is verplicht om na ontvangst van elke byte een bevestiging (acknowledge) te genereren. De master genereert dan een extra clockpuls die bij het acknowledge-bit hoort.

De schakeling die een acknowledge geeft, moet de SDA-lijn gedurende de acknowledge-clockpuls LAAG trekken.

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.6-6: De bus-condities.

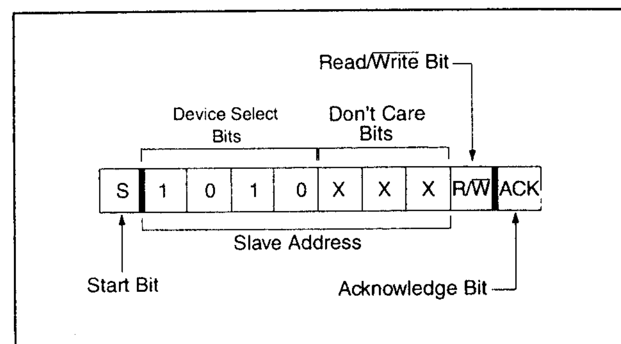


Figuur 7/6.5.6-7: Acknowledge-timing.

Hierbij moeten uiteraard de setup- en houdtijden in rekening worden gebracht, zodat de SDA-lijn gedurende deze clockpuls stabiel LAAG is. Een master moet aan de slaaf een "end of data" signaleren door na de laatste byte geen acknowledge-bit te genereren. In dit geval moet de slaaf de datalijn HOOG laten, zodat de master een STOP-conditie kan genereren (zie figuur 7/6.5.6-7).

Device-adressering

Nadat een START-conditie is gegenereerd verstuurt de master een besturingsbyte, bestaande uit een slaaf-adres en een Read/$\overline{\text{Write}}$-bit dat aangeeft welk type operatie moet worden uitgevoerd. Het slaafadres bestaat uit een 4 bit device-code (1010), gevolgd door drie don't care bits (zie figuur 7/6.5.6-8). Het laatste bit van het control-byte bepaalt welke operatie moet worden uitgevoerd (bij "1" wordt een lees-operatie geselecteerd, bij "0" wordt geschreven). De bus wordt voortdurend gecontroleerd op slaaf-adres. Als het slaaf-adres "waar" en de schakeling niet in de programmeer-mode is, wordt een acknowledge gegenereerd.



Figuur 7/6.5.6-8: Formaat van het control-byte.

Schrijf-operaties

Byte-Write: na het START-signaal van de master worden de device-code (4 bits), de don't care bits (3 bits) en het R/$\overline{\text{W}}$-bit (dat logisch LAAG is) door de master-transmitter op de bus gezet. Dit laat aan de geadresseerde slaaf-receiver weten dat een byte met

6.5 PIC-typen 8 bit microcontrollers

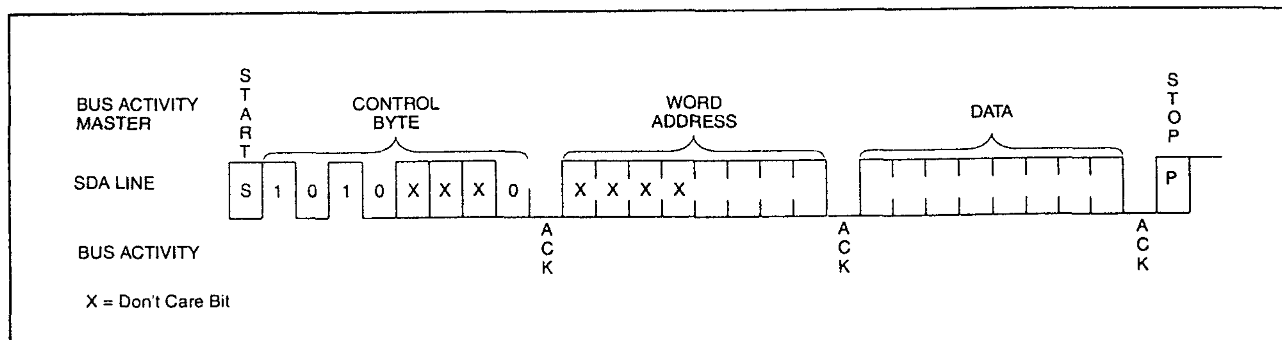
een woord-adres zal volgen nadat hij - gedurende de 9e clockcyclus - een acknowledge-bit heeft gegenereerd. Daarom is het woord-adres het volgende byte dat door de master wordt verstuurd. Dit wordt in de adres-pointer geschreven. Alleen de laagste vier adresbits worden door de schakeling gebruikt; de hoogste vier bits zijn don't cares. Op het adres-byte moet een acknowledge volgen en de master zendt daarop het datawoord, bestemd voor de geadresseerde geheugenplaats uit. Het geheugen bevestigt nogmaals en de master genereert een STOP-conditie. Hierdoor wordt de interne schrijfcyclus geïnitialiseerd en gedurende deze tijd worden er geen acknowledge-signalen gegenereerd (figuur 7/6.5.6-9).
Na een byte-write commando wordt de interne adresteller niet verhoogd en zal deze dus naar dezelfde adres-lokatie wijzen die net werd beschreven. Als ergens binnen de schrijfcommando-cyclus een stopbit wordt verzonden voordat de gehele volgorde klaar is, wordt er gestopt met schrijven. Als er meer dan 8 bits zijn verzonden voordat het stopbit optreedt, zal de schakeling het daarvoor geladen byte clearen en opnieuw met het laden van de data-buffer beginnen. Als meer dan één databyte werd verzonden en het stopbit verschijnt voordat een hele groep van acht databits werd verstuurd, zal het schrijfcommando afhaken en wordt geen data geschreven. Het EEPROM-geheugen maakt gebruik van een V_{CC}-drempeldetector die de interne wis/schrijf-logica afschakelt als V_{CC} beneden de minimumwaarde V_{DD} is gekomen.

Acknowledge Polling

Aangezien de schakeling geen acknowledge verstuurt tijdens een schrijfcyclus, kan dit worden gebruikt om te bepalen of de cyclus is volbracht (dit wordt gebruikt om de bussnelheid te verhogen). Zodra een STOP-conditie of een schrijfcommando door de master is afgegeven, initieert de schakeling de *intern getimede schrijfcyclus*. ACK polling kan direct worden geïnitialiseerd. Dit houdt in dat de master een START-conditie verstuurt, gevolgd door het control-byte voor een schrijfcommando (R/$\overline{E}$ = "0"). Als de schakeling nog bezig is met de schrijfcyclus wordt geen ACK afgegeven. Als er geen ACK wordt gegeven moeten het START-bit en het control-byte opnieuw worden verstuurd. Is de cyclus volbracht dan zal de schakeling wel een ACK afgeven en kan de master doorgaan met het volgende lees- of schrijfcommando (zie figuur 7/6.5.6-10).

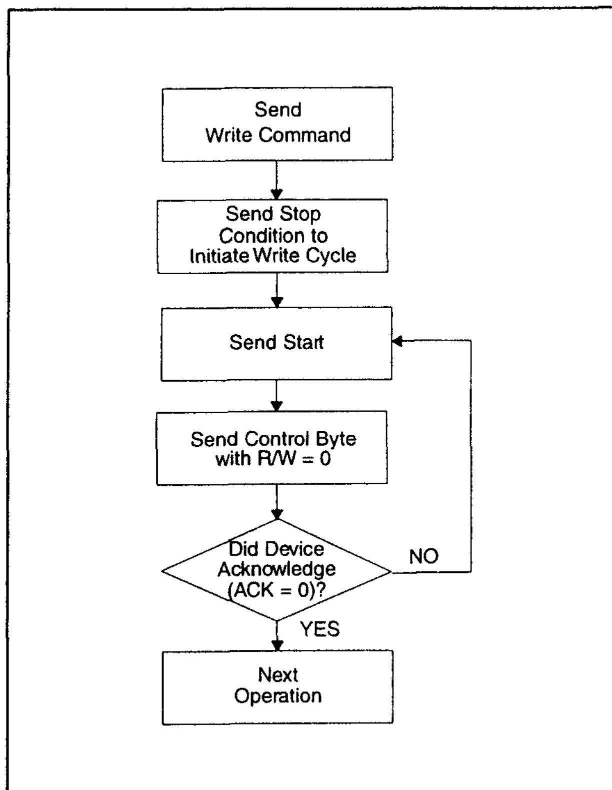
Lees-operaties

Lees-operaties worden op dezelfde manier geïnitialiseerd als schrijf-operaties, maar dan met een op "1" gezet R/$\overline{W}$-bit. Er zijn in principe drie typen lees-operaties:
- Current Address Read;
- Random Read;
- Sequential Read.



Figuur 7/6.5.6-9: Byte Write.

6.5 PIC-typen 8 bit microcontrollers



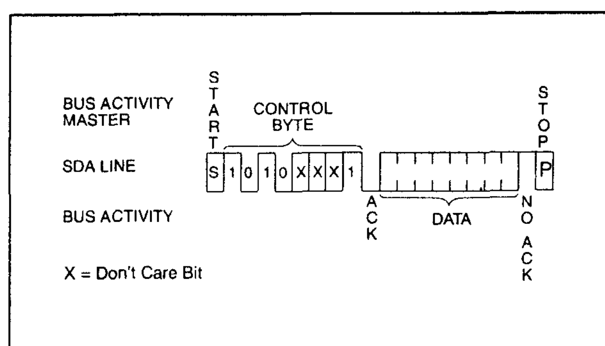
Figuur 7/6.5.6-10: Flow-diagram van de Acknowledge Polling.

Current Address Read

Dit bevat een adresteller die het adres van het laatst bezochte woord bevat (intern met één opgehoogd). Als het voorafgaand uitgelezen adres dus n was, wordt het volgende lopende adres uitgelezen op n+1. Na ontvangst van het slaafadres (met het R/$\overline{W}$-bit op "1") geeft de schakeling een acknowledge en verstuurt het 8 bit datawoord.
De master zal de overdracht niet bevestigen, maar genereert een STOP-conditie (figuur 7/6.5.6-11).

Sequential Read

Opeenvolgende (sequential) lees-operaties worden op dezelfde manier geïnitialiseerd als een random read, maar nu levert de master een acknowledge nadat de schakeling het eerste data-byte heeft verstuurd, in plaats van een stop-conditie zoals bij random read. Hierdoor gaat de schakeling het volgende aansluitend geadresseerde 8 bit woord versturen (figuur 7/6.5.6-13).
Om opeenvolgende leesoperaties te kunnen uitvoeren is er een interne adrespointer aanwezig die na elke lees-operatie met één wordt verhoogd. Deze adrespointer maakt het mogelijk om de gehele geheugeninhoud in één operatie serieel uit te lezen.

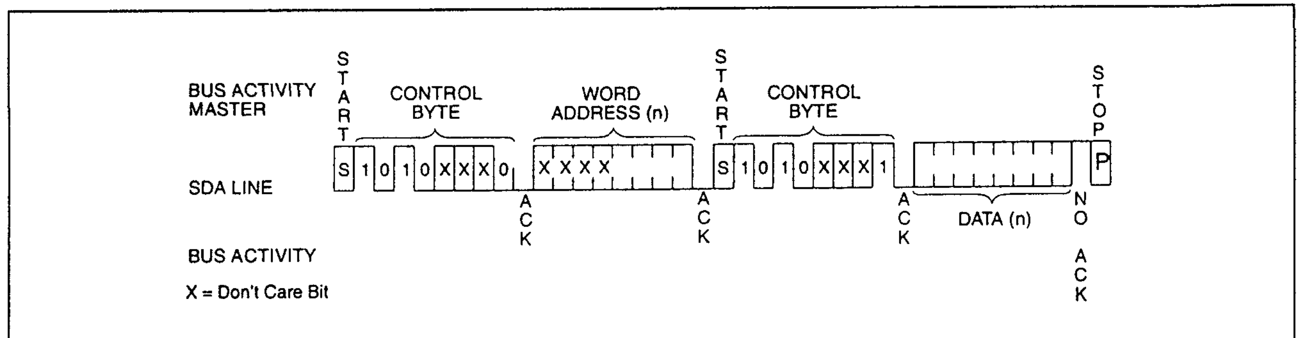


Figuur 7/6.5.6-11: Current Address Read.

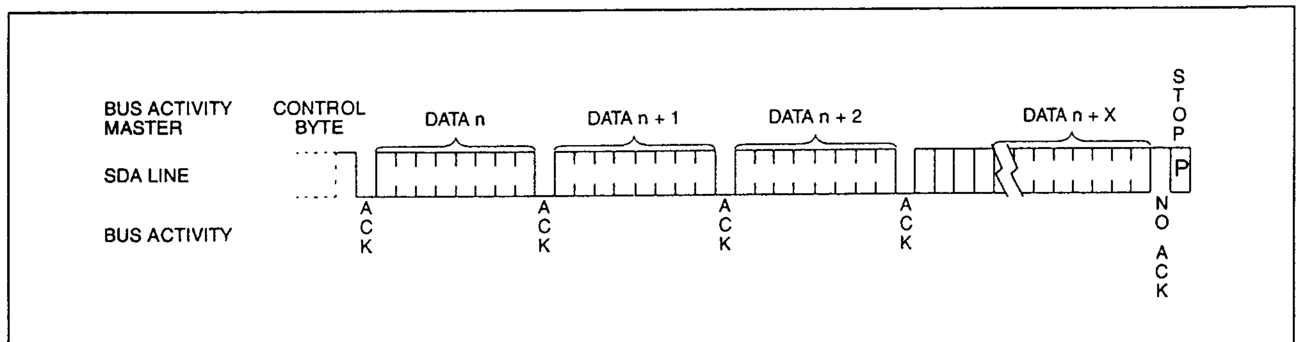
Random Read

In de Random Read mode kan de master elke willekeurige plaats in het geheugen bereiken. Om dit type lees-operatie te kunnen uitvoeren, moet eerst het woord-adres worden vastgesteld. Dit wordt gedaan door het woord-adres als deel van een schrijf-operatie naar de schakeling te versturen. Zodra het woord-adres is verstuurd genereert de master na de acknowledge een START-conditie. Hierdoor wordt de schrijf-operatie beëindigd, maar niet voordat de interne adrespointer is gezet. Dan levert de master nogmaals het control-byte, maar dan met het R/$\overline{W}$-bit op "1". Hierna levert hij een acknowledge en verstuurt het 8 bit datawoord. De master zal de overdracht niet bevestigen, maar genereert in plaats daarvan een STOP-conditie, waarna de overdracht wordt gestopt (figuur 7/6.5.6-12). Na dit commando wijst de interne adresteller naar het adres dat na de zojuist gelezen locatie komt.

6.5 PIC-typen 8 bit microcontrollers



Figuur 7/6.5.6-12: Random Read.



Figuur 7/6.5.6-13: Sequential Read.

Instructieset

Inleiding

Ook elke PIC12CE5xx-instructie is een 12 bit woord, gesplitst in een OPCODE (die de soort instructie specificeert) en één of meer OPERANDS die de operatie van de instructie verder specificeren.

Tabel 7/6.5.5-13 geeft een samenvatting van de byte- en bit-georiënteerde instructies en letterlijke en besturings operaties, terwijl in tabel 7/6.5.5-14 de opcode field-beschrijvingen worden getoond.

- In byte-georiënteerde instructies wordt met "f" een file register-designator bedoeld en met "d" een bestemmings-(destination-) designator.
- In bit-georiënteerde instructies is "b" een bit-field designator die het nummer van het bit selecteert dat door de operatie wordt beïnvloed. Hierbij is "f" het nummer van de file waarin het bit zich bevindt.
- In letterlijke en besturings operaties, geeft "k" een 8 of 9 bit constante of een letterlijke waarde aan.

Alle instructies worden in één enkele instructiecyclus uitgevoerd, tenzij een conditionele test "waar" is of als de programcounter als gevolg van een instructie is veranderd. In dat geval zijn voor de uitvoering twee instructie-cycli nodig.

Een instructie bestaat uit vier oscillatorperioden.

Bij een oscillator-frequentie van 4 MHz is de normale uitvoeringstijd van een instructie dus 1 μs.

In tabel 7/6.5.5-15 zijn de drie algemene formaten te zien die de instructies kunnen hebben.

Voor alle voorbeelden is hetzelfde formaat gebruikt om een hexadecimaal getal voor te stellen: 0xhhh (waarin "h" een hexadecimale digit voorstelt).

6.5 PIC-typen 8 bit microcontrollers

```
    TITLE "PIC with EEPROM Data Memory Interface"
    LIST P=12CE518                     ; Change to 12CE519 if using PIC12CE519
#include <p12CE518.inc>
;
;       Program:          EEPROM.ASM
;       Revision Date:
;                         10-10-97      Adapted to 12CE51x parts
;
; PIC12CE51X EEPROM communication code.  This code should be linked in
; with the application.  These routines provide the following functionality:
; write byte random address
; read byte random address
; read byte next address
;
; read sequential is not supported.
;
; If the operation is successful, bit 7 of PC_OFFSET will be set, and
; the functions will return W=1.  If the memory is busy with a write
; cycle, it will not ACK the command.  The functions will return with
; bit 7 of PC_OFFSET cleared and and W will be set to 0.
;
; Based on Franco code.
;
; Must reside on the lower half of code page (address 0-FF).
;
; This provides users with highly compressed assembly code for
; communication between the EEPROM and the Microcontroller, which
; leaves a maximum amount of code space for the core application.
;
; NOPs have been added to meet the timing specs for the memory at 4 MHz
; and low voltage.  Applications running at slower clock rates and those
; operating within 4.5-5.5V may be able to remove some of the NOPs.
;
; This code is specifically written for the interface hardware of the
; 12CE51x parts.  See AN571 for the unmodified routines.
;***************************************************************************
;************************* EEPROM Subroutines ******************************
;***************************************************************************
; Communication for EEPROM based on I2C protocol, with Acknowledge.
;
; Byte_Write: Byte write routine
;       Inputs:     EEPROM Address      EEADDR
;                   EEPROM Data         EEDATA
;       Outputs:    Return 01 in W if OK, else return 00 in W
;
; Read_Current:     Read EEPROM at address currently held by EE device.
;       Inputs:     NONE
;       Outputs:    EEPROM Data         EEDATA
;                   Return 01 in W if OK, else return 00 in W
;
; Read_Random:      Read EEPROM byte at supplied address
;       Inputs:     EEPROM Address      EEADDR
;       Outputs:    EEPROM Data         EEDATA
;                   Return 01 in W if OK, else return 00 in W
;
; Note: EEPROM subroutines will set bit 7 in PC_OFFSET register if the
;       EEPROM acknowledged OK, else that bit will be cleared.  This bit
;       can be checked instead of refering to the value returned in W
;***************************************************************************
;
; OPERATION:
;     Byte Write:
;       load EEADDR and EEDATA
;       then CALL BYTE_WRITE
;
;     Read Random:
;       Load EEADDR
;       then CALL READ_RANDOM
;       data read returned in EEDATA
;
;     Read Current
;       no setup necessary
;       CALL READ_CURRENT
;       data read returned in EEDATA
;
;***************************************************************************
;
; These functions consume:
; 77 words Programming Memory
; 5 file registers which are overlayable.  That is, they can share with
; other functions as long as they are mutually exclusive in time.  See
; udata_ovr in the linker manual.
; 1 stack level (the call to the function itself.  These functions do not
; call any lower level functions).
;
;
```

Figuur 7/6.5.-14a: Voorbeeld-code voor lezen/schrijven naar het EEPROM data-geheugen.

6.5 PIC-typen 8 bit microcontrollers

```
;******************************************************************
;************************* Variable Listing *************************
;******************************************************************
OK              EQU     01H
NO              EQU     00H

I2C_PORT        EQU     GPIO            ; Port B control register, used for I2C
SCL             EQU     07H             ; EEPROM Clock, SCL (I/O bit 7)
SDA             EQU     06H             ; EEPROM Data,  SDA (I/O bit 6)

EE_OK           EQU     07H             ; Bit 7 in PC_OFFSET used as OK flag for EE

        udata_ovr
PC_OFFSET       RES     1               ; PC offset register (low order 4 bits),
                                        ;  value based on operating mode of EEPROM.
                                        ;  Also, bit 7 used for EE_OK flag
EEADDR          res     1               ; EEPROM Address
EEBYTE          res     1               ; Byte sent to or received from
                                        ; EEPROM (control, address, or data)
COUNTER         res     1               ; Bit counter for serial transfer

        udata
EEDATA          res     1               ; EEPROM Data

    global      READ_CURRENT
    global      READ_RANDOM
    global      WRITE_BYTE
    global      EEADDR
    global      EEDATA
    global      PC_OFFSET


;******************** Set up EEPROM control bytes ********************
;******************************************************************
        code
READ_CURRENT
    MOVLW       B'10000100'         ; PC offset for read current addr.  EE_OK bit7='1'
    MOVWF       PC_OFFSET           ; Load PC offset
    GOTO        INIT_READ_CONTROL

WRITE_BYTE
    MOVLW       B'10000000'         ; PC offset for write byte.  EE_OK: bit7 = '1'
    GOTO        INIT_WRITE_CONTROL

READ_RANDOM
    MOVLW       B'10000011'         ; PC offset for read random.  EE_OK: bit7 = '1'

INIT_WRITE_CONTROL
    MOVWF       PC_OFFSET           ; Load PC offset register, value preset in W
    MOVLW       B'10100000'         ; Control byte with write bit, bit 0 = '0'

START_BIT
    BCF         I2C_PORT,SDA        ; Start bit, SDA and SCL preset to '1'


;******* Set up output data (control, address, or data) and counter *******
;******************************************************************
PREP_TRANSFER_BYTE
    MOVWF       EEBYTE              ; Byte to transfer to EEPROM already in W
    MOVLW       .8                  ; Counter to transfer 8 bits
    MOVWF       COUNTER
;*********** Clock out data (control, address, or data) byte ***********
;******************************************************************
OUTPUT_BYTE
    BCF     I2C_PORT,SCL            ; Set clock low during data set-up
    RLF     EEBYTE, F               ; Rotate left, high order bit into carry bit
    BCF     I2C_PORT,SDA            ; Set data low, if rotated carry bit is
    SKPNC                           ;   a '1', then:
    BSF     I2C_PORT,SDA            ; reset data pin to a one, otherwise leave low
    NOP
    BSF     I2C_PORT,SCL            ; clock data into EEPROM
    DECFSZ  COUNTER, F              ; Repeat until entire byte is sent
    GOTO    OUTPUT_BYTE
    NOP                             ; Needed to meet Timing (Thigh=4000nS)

;************************ Acknowkedge Check ************************
;******************************************************************
    BCF     I2C_PORT,SCL            ; Set SCL low, 0.5us < ack valid < 3us
    NOP                             ; Needed to meet Timing (Tlow= 4700nS)
    BSF     I2C_PORT,SDA
    GOTO    $+1                     ;
;   NOP                             ; Necessary for SCL Tlow at low voltage,
;   NOP                             ; Tlow=4700nS
    BSF     I2C_PORT,SCL            ; Raise SCL, EEPROM acknowledge still valid
    BTFSC   I2C_PORT,SDA            ; Check SDA for acknowledge (low)
    BCF     PC_OFFSET,EE_OK         ; If SDA not low (no ack), set error flag
    BCF     I2C_PORT,SCL            ; Lower SCL, EEPROM release bus
    BTFSS   PC_OFFSET,EE_OK         ; If no error continue, else stop bit
    GOTO    STOP_BIT
```

Figuur 7/6.5.-14b: Voorbeeld-code voor lezen/schrijven naar het EEPROM data-geheugen (vervolg).

6.5 PIC-typen 8 bit microcontrollers

```
;*****  Set up program counter offset, based on EEPROM operating mode  *****
;**************************************************************************
        MOVF    PC_OFFSET,W
        ANDLW   B'00001111'
        ADDWF   PCL, F
    GOTO    INIT_ADDRESS          ;PC offset=0, write control done, send address
    GOTO    INIT_WRITE_DATA       ;PC offset=1, write address done, send data
    GOTO    STOP_BIT              ;PC offset=2, write done, send stop bit
    GOTO    INIT_ADDRESS          ;PC offset=3, write control done, send address
    GOTO    INIT_READ_CONTROL     ;PC offset=4, send read control
    GOTO    READ_BIT_COUNTER      ;PC offset=5, set counter and read byte
    GOTO    STOP_BIT              ;PC offset=6, random read done, send stop

;**********  Initalize EEPROM data (address, data, or control) bytes  ******
;**************************************************************************
INIT_ADDRESS
    INCF    PC_OFFSET, F          ; Increment PC offset to 2 (write) or to 4 (read)
    MOVF    EEADDR,W              ; Put EEPROM address in W, ready to send to EEPROM
    GOTO    PREP_TRANSFER_BYTE


INIT_WRITE_DATA
    INCF    PC_OFFSET, F          ; Increment PC offset to go to STOP_BIT next
    MOVF    EEDATA,W              ; Put EEPROM data in W, ready to send to EEPROM
    GOTO    PREP_TRANSFER_BYTE

INIT_READ_CONTROL
    BSF     I2C_PORT,SCL          ; Raise SCL
    BSF     I2C_PORT,SDA          ; raise SDA
    INCF    PC_OFFSET, F          ; Increment PC offset to go to READ_BIT_COUNTER next
    MOVLW   B'10100001'           ; Set up read control byte, ready to send to EEPROM
    GOTO    START_BIT             ;   bit 0 = '1' for read operation
;*************************  Read EEPROM data  ***************************
;**************************************************************************
READ_BIT_COUNTER
    BSF     I2C_PORT,SDA          ; set data bit to 1 so we're not pulling bus down.
    NOP
    BSF     I2C_PORT,SCL
    MOVLW   .8                    ; Set counter so 8 bits will be read into EEDATA
    MOVWF   COUNTER

READ_BYTE
    BSF     I2C_PORT,SCL          ; Raise SCL, SDA valid.  SDA still input from ack
    SETC                          ; Assume bit to be read = 1
    BTFSS   I2C_PORT,SDA          ; Check if SDA = 1
    CLRC                          ; if SDA not = 1 then clear carry bit
    RLF     EEDATA, F             ; rotate carry bit (=SDA) into EEDATA;
    BCF     I2C_PORT,SCL          ; Lower SCL
    bsf     I2C_PORT,SDA          ; reset SDA
    DECFSZ  COUNTER, F            ; Decrement counter
    GOTO    READ_BYTE             ; Read next bit if not finished reading byte

    BSF     I2C_PORT,SCL
    NOP
    BCF     I2C_PORT,SCL
;*****************  Generate a STOP bit and RETURN  *********************
;**************************************************************************
STOP_BIT
    BCF     I2C_PORT,SDA          ; SDA=0, on TRIS, to prepare for transition to '1'
    BSF     I2C_PORT,SCL          ; SCL = 1 to prepare for STOP bit
    GOTO    $+1                   ; equivalent 4 NOPs neccessary for I2C spec Tsu:sto = 4.7us
    GOTO    $+1
    BSF     I2C_PORT,SDA          ; Stop bit, SDA transition to '1' while SCL high

    BTFSS   PC_OFFSET,EE_OK       ; Check for error
    RETLW   NO                    ; if error, send back NO
    RETLW   OK                    ; if no error, send back OK

;**************************************************************************
;***********************  End EEPROM Subroutines  ************************
        end
```

Figuur 7/6.5.-14c: Voorbeeld-code voor lezen/schrijven naar het EEPROM data-geheugen (vervolg).

Elektrische en timing-eigenschappen

In de figuren 7/6.5.5-40 tot en met -44 en de tabellen 7/6.5.5-16 tot en met -24 wordt een overzicht gegeven van de elektrische- en timing eigenschappen van de PIC12C5xx- en PIC12CE5xx-families.

Appendix

De volgende routines zijn geschreven voor operaties bij 4 MHz, waarbij de "worst case timing" optreedt. Deze routines kunnen zonder wijzigingen bij ook lagere frequenties worden gebruikt. Bij veel lagere frequenties kunnen sommige NOP's worden verwijderd om de code-omvang te beperken.

SDA en SCL

De EEPROM-interface is een 2-draads busprotocol, bestaande uit data (SDA) en een clock (SCL). Hoewel deze lijnen naar het GRD-register zijn gemapt, zijn ze niet bereikbaar als externe pennen. Het werken met SDA en SCL is ook iets afwijkend van GP0 tot en met GP5, zoals hieronder te zien is. Om namelijk overtollige code bij het modificeren van de TRIS te vermijden zijn SDA en SCL altijd uitgangen. Om data uit de EEPROM te kunnen lezen, moet een "1" op SDA worden gezet, waardoor deze lijn in de hoogimpedante toestand komt. Hierbij is alleen de interne 100 kΩ optrekweerstand aktief op de SDA-lijn.

- SDA:
  Ingebouwde 100 kΩ optrekweerstand naar V_{DD}.
  Open-drain (alleen pull-down)
  Altijd uitgang, onafhankelijk van TRIS
  Geeft een "1" bij resetten.
- SCL:
  Echte CMOS uitgang
  Altijd uitgang, onafhankelijk van TRIS
  Geeft een "1" bij resetten.

Bij de volgende voorbeelden is nodig:

- Code-ruimte: 77 woorden
- RAM-ruimte:
  5 bytes (waarvan 4 overlayable)
- Stack-niveaus:
  1 (alleen de CALL-functie zelf)
- Timing:
  WRITE_BYTE: 328 cycli
  READ_CURRENT: 212 cycli
  READ_RANDOM: 416 cycli
- I/O-pennen:
  0 (geen externe I/O).

Deze code moet in de laagste helft van een pagina worden opgeslagen. De kleine afmetingen van de code zijn een gevolg van een aftast-tabel (een lijst van procedures die op volgorde moeten worden opgeroepen). De tabel maakt gebruik van een ADDWF PCL,F instructie (die een 8 bit adres nodig heeft, waardoor de code zich in de eerste 256 adressen van een pagina moeten bevinden).

6.5 PIC-typen 8 bit microcontrollers

7/7

Processoren voor fuzzy logic

Inhoud

7/7.1

Achtergrond-informatie

Inleiding

Begrenzingen van harde logica

Met de toenemende digitalisering van de elektronica wordt men steeds vaker geconfronteerd met het gegeven dat bepaalde processen niet zo gemakkelijk te vangen zijn in harde logica. Ingewikkelde regelsystemen, die vroeger met behulp van analoge schakelingen (de zogenoemde analoge computers) werden uitgevoerd, worden steeds vaker volledig digitaal geregeld met computers. Maar die computers eisen wél keiharde logica, zowel in hun invoergegevens als in hun beslissingen! Deze beslissingsregels, waarmee de computer uit de invoergegevens bepaalde uitvoergegevens afleidt, moeten volledig en ondubbelzinnig bepaald zijn. Met de beslissingsregel *"als het glas in de oven iets vloeibaarder wordt dan uit ervaring wenselijk is, dan moet de gaskraan ietsjes dicht gedraaid worden"* kan een computer helemaal niets beginnen. Natuurlijk zou een dergelijk probleem nog wel omgezet kunnen worden in harde logica door de temperatuur van het glas als maatstaf te nemen voor de vloeibaarheid ervan, de temperatuur analoog te meten, om te zetten in harde logische digitale codes en deze door een aantal harde beslissingsregels te laten evalueren. Dergelijke Boolse vergelijkingen leveren dan weer harde digitale uitgangscodes op, die weer omgezet kunnen worden in analoge signalen die gaskranen dicht kunnen draaien.
Maar bij zeer ingewikkelde industriële regelingen, waar tientallen ingangsgrootheden invloed hebben op een proces en bovendien een belangrijke factor, namelijk *de ervaring van de menselijke procesbegeleider* heel veel invloed heeft op het eindresultaat, komt men snel in de problemen.

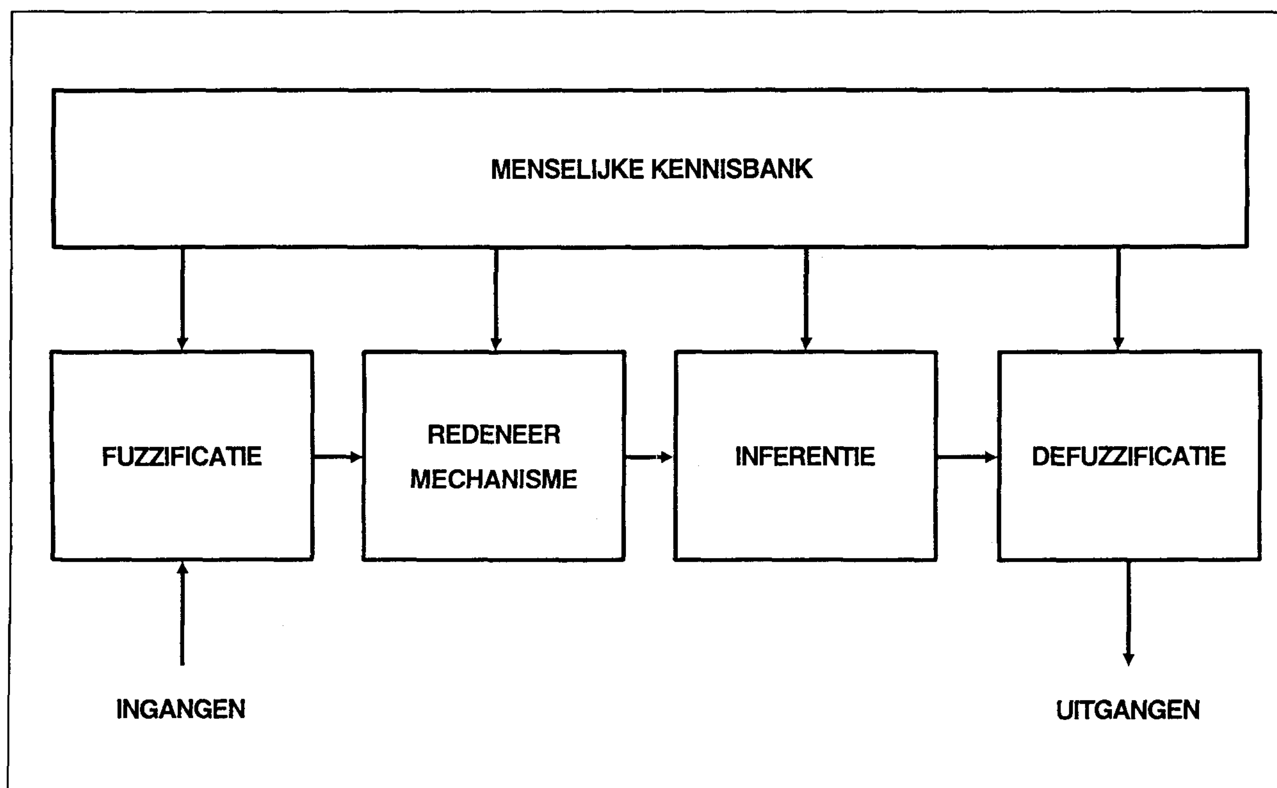
Vage logica

Vandaar dat in de jaren zeventig een geheel nieuwe benadering van dergelijke problemen werd gezocht. Alle elementen van een verzameling worden nu niet in twee keiharde set's ingedeeld ("ja" en "nee") maar volgens een glijdende schaal van "0 % ja" (en dus "100 % nee" tot "100 % ja" (en dus "0 % nee"). Een dergelijke benadering sluit veel beter aan bij de manier waarop mensen informatie classificeren.
Nu mag men niet de vergissing begaan om te denken dat in- en uitvoergegevens niet meer feitelijk "hard" zijn! Er is weinig vaags aan de vage logica! De kunst van het principe is de in- en uitgangsgegevens te vangen in zogenoemde "lidmaatschapsfuncties" en nadien beslissingsregels op te stellen waarmee het systeem uit de voeten kan.

Lotfi Zadeh

De fundamenten van de vage logica zijn, zoals tegenwoordig met zovele technieken het geval is, terug te vinden in de theoretische wiskunde. In 1965 publiceerde de wiskundige L. A. Zadeh van de Berkeley universiteit in de Verenigde Staten een boek, getiteld "Fuzzy Sets in Information and Control". De bedoeling was een wiskundige theorie op te stellen, waarmee taalkundige stellingen vertaald konden worden naar wiskundige formules.

7.1 Achtergrond-informatie



Figuur 7/7.1-1: De vier voornaamste stappen van een fuzzy-proces samengevat.

Het was Zadeh onmiddellijk duidelijk dat dit alleen maar kon als werd afgeweken van de klassieke verzamelingsleer, waarbij een element ondubbelzinnig tot een set behoort. Vandaar voerde hij het begrip "lidmaatschapsgraad" in, waarmee wordt aangegeven in hoeverre een bepaald element van de verzameling tot een bepaalde set behoort. Deze lidmaatschapsgraad kan dan worden omgezet in een lidmaatschapsfunctie, afgekort tot LF, in wezen niets anders dan een grafische voorstelling van de graad van lidmaatschap van ieder element uit de verzameling bij een bepaalde set. Op deze manier worden alle ingangsvariabelen omgezet in een of meerdere LF's. Dit proces noemt men de "fuzzificatie". Ook de uitgangsvariabelen moeten op een identieke manier gefuzzificeerd worden. Nadien moeten bepaalde relaties tussen deze in- en uitgangsgrootheden worden opgesteld. Dit noemt men het "redeneer mechanisme". De regels voor dit redeneer mechanisme worden ontleend uit de kennis die de bedenker van het systeem heeft. Vandaar dat vaak wordt gerefereerd naar de "kennisbank" en de regels "ervaringsregels" worden genoemd. Dit proces gebeurt in regels, die welbekend zijn uit de "harde" logica. De meest algemene structuur van een dergelijke regel is:

ALS

aan bepaalde voorwaarden voldaan wordt

DAN

moet op een bepaalde manier een actie ondernomen worden.

Het enige verschil is dat de voorwaarden en de acties niet alleen "ja" en "nee" kunnen zijn, maar ook tussenwaarden kunnen hebben. De laatste stap noemt men de "defuzzificatie". Hierin worden de LF's van de ingangsgrootheden en van de uitgangsgrootheden gekoppeld aan de ervaringsregels, met als gevolg dat er bepaalde uitgangsfuncties ontstaan.

Het fuzzy-proces

Het volledige proces van fuzzy logic kan dus voorgesteld worden zoals samengevat in figuur 7/7.1-1:

- Fuzzificatie:
  Ingangsgrootheden en uitgangsgrootheden worden verwerkt tot lidmaatschapsfuncties, waarbij rekening wordt gehouden met ervaringsregels die in de kennisbank aanwezig zijn.
- Redeneer mechanisme:
  Hierbij is het de bedoeling een aantal ALS ... DAN regels op te stellen waarbij de vage ingangsvariabelen gekoppeld worden aan een of meerdere vage uitgangsvariabelen. Deze regels beschrijven het proces waarop het fuzzy-proces wordt toegepast.
- Inferentie:
  De inferentie is de techniek van het redeneermechanisme, waarmee de fuzzy-processor uit de opgestelde regels een bepaalde uitgangsfunctie afleidt.
- Defuzzifucatie:
  Uit de regels worden waarden voor de uitgangsvariabelen afgeleid, die niet langer vaag zijn maar concreet, zodat elektronische schakelingen er weer iets mee kunnen aanvangen.

Specifieke toepassingsterreinen van vage logica

Vage logica zal vooral worden toegepast in die processen, die met "harde" logica heel moeilijk te omschrijven zijn. Een typisch voorbeeld van een dergelijk proces is patroonherkenning. Patroonherkenning staat in het middelpunt van de belangstelling, omdat technieken die in staat zijn met een aan zekerheid grenzende waarschijnlijkheid een bepaald patroon te herkennen een zeer grote toekomst hebben. Patroonherkenning vormt immers het kloppend hart van zeer uiteenlopende praktisch toepassingen zoals:

- elektronische handschriftherkenning;
- elektronische spraakherkenning;
- verminkte gegevens terugwinnen uit beschadigde of gestoorde data-stromen;
- automatische sorteersystemen, zoals het sorteren van tomaten op grootte, vorm of rijpheid;
- automatische visuele inspectie door middel van een video-camera van de producten op een lopende band.

Al deze toepassingen hebben een belangrijke eigenschap gemeen en dat is dat de ingangsvariabelen niet exact omschreven kunnen worden. Het heeft geen zin om te proberen vorm en kleur van een tomaat wiskundig te beschrijven in exacte formules. Ieder tomaat zal immers in min of meerdere mate van een dergelijke absolute beschrijving afwijken! Hier kunnen alleen de vage beslissingen van de vage logica bruikbare oplossingen bieden!

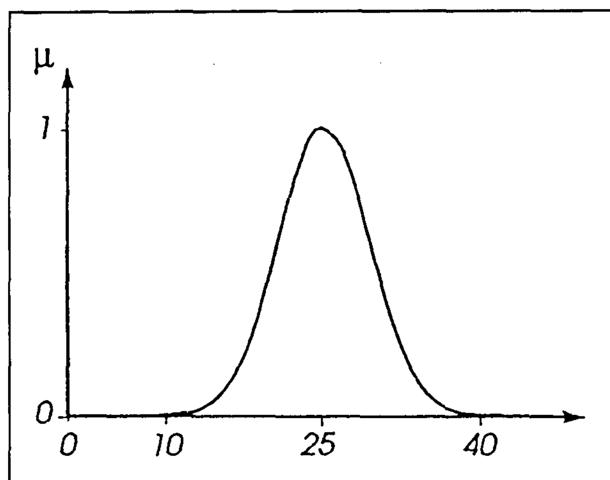
De fuzzificatie

Het principe van de lidmaatschapsfunctie

Zoals reeds gesteld is het voornaamste instrument van de vage logica de "lidmaatschapsfunctie LF", in het Engels "membership function MF" genoemd en in het Duits "Zugehörigkeitsfunktion ZK". Soms wordt in het Nederlands ook de term "toebehorensfunctie" gebruikt. Het woord lidmaatschap dekt echter veel beter de lading en vandaar dat deze in dit hoofdstuk gebruikt zal worden. De lidmaatschapsfunctie is een grafiek die aangeeft in hoeverre een bepaald element uit een verzameling tot een bepaalde set behoort. Een voorbeeldje zal dit verduidelijken. Stel dat men aan duizend personen de vraag stelt aan welke leeftijd men denkt als men het begrip "een jonge man" in gedachten neemt. Waarschijnlijk zal er één leeftijd het vaakst genoemd worden, bijvoorbeeld "25 jaar". Rond deze vaakst genoemde leeftijd zullen antwoorden liggen die iets minder vaak genoemd worden, zoals "24 jaar" en "26 jaar". Tot slot zullen er ook antwoorden zijn, die maar door een paar mensen genoemd worden, zoals "40 jaar" en "10 jaar". Men kan nu aan de meest genoemde leeftijd

7.1 Achtergrond-informatie

een waarde 1 toekennen en alle andere leeftijden hieraan relateren. Als het antwoord "25 jaar" 120 keer genoemd wordt en het antwoord "15 jaar" 12 keer, dan relateert men het antwoord "15 jaar" aan het antwoord "25 jaar" door er de waarde 0,1 aan toe te kennen.
Men is nu klaar voor het opstellen van de lidmaatschapsfunctie voor het begrip "een jonge man". Die functie bestaat uit de grafiek die getekend is in figuur 7/7.1-2. Op de verticale as worden alle aan het antwoord "25 jaar" gerelateerde getallen uitgezet, op de horizontale as alle verkregen antwoorden. Het resultaat is een Gaussiaanse verdelingscurve.



Figuur 7/7.1-2: De lidmaatschapsfunctie voor het begrip "een jonge man".

Een lidmaatschapsfunctie heeft dus steeds een vertikale as die van 0 tot 1 loopt. De indeling en schaal van de horizontale as is afhankelijk van het verschijnsel dat men wil fuzzificeren.

Nog wat nieuwe begrippen

Aan de hand van de definitie van een lidmaatschapsfunctie kunnen nog enige nieuwe begrippen gedefinieerd worden:
- quantificatie;
- lidmaatschapsgraad;
- fuzzy-set;
- linguïstische variabele;
- tolerantie.

Quantificatie

Wat in figuur 7/7.1-2 gebeurt is, is dat een vaag begrip als "een jonge man" *gequantificeerd* werd. Er worden waarden aan toegekend, waardoor het voor een marsmannetje mogelijk zou zijn een relatie te leggen tussen leeftijden van aardbewoners en het begrip "een jonge man", zelfs zonder dat hij ooit een aardbewoner had gezien.

Lidmaatschapsgraad

Aan de hand van de lidmaatschapsfunctie kan men aan iedere leeftijd een zogenoemde *lidmaatschapsgraad* toekennen. Deze graad wordt standaard voorgesteld door μ. In het voorbeeld is de lidmaatschapsgraad van het antwoord "25 jaar" gelijk aan 1. Het zal duidelijk zijn dat de waarde van μ steeds begrensd wordt door het minimum 0 en het maximum 1.

Fuzzy-set

Het volledig bereik van de lidmaatschapsfunctie noemt men de *fuzzy-set*. Figuur 7/7.1-2 geeft dus de fuzzy-set voor het vage begrip "een jonge man".

Linguïstische variabele

Het vage begrip, waarvoor men een fuzzy-set opstelt, wordt *linguïstische variabele* genoemd. Het is een variabele, omdat er verschillende waarden van μ uit afgeleid kunnen worden. Het is een linguïstische variabele, omdat de quantificering ervan niet wiskundig exact is, maar taalkundig gedefinieerd wordt.

Vereenvoudiging en tolerantie

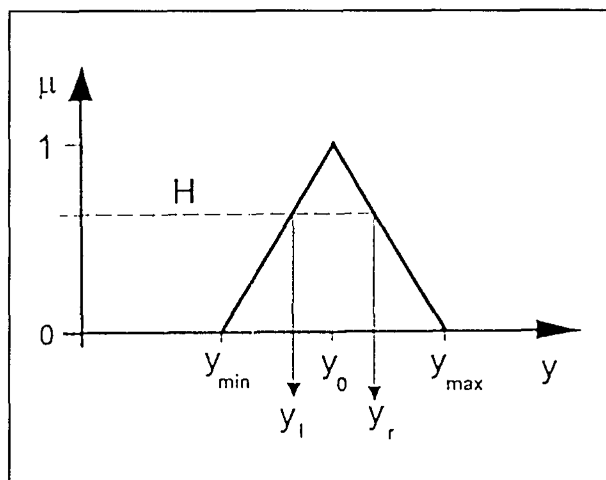
Ook in de vage logica moet gerekend worden, want er moeten immers uitgangsgrootheden worden afgeleid uit de ingangsgrootheden. Rekenen aan ingangsgrootheden kan alleen als het verloop van deze grootheden in exacte wiskundige formules kan worden vast gelegd. Met andere woorden: aan iedere lidmaatschapsfunctie moet een wis-

kundige vergelijking gekoppeld worden, die de functie zo goed mogelijk beschrijft. De algemene vorm van een dergelijke vergelijking is:

$\mu A_{(x)}$ = wiskundige uitdrukking

waarbij x een bepaald element uit de vage verzameling is van de linguïstische variabele A. In het voorbeeld zou x bijvoorbeeld gelijk kunnen zijn aan "21 jaar", terwijl A natuurlijk gelijk is aan "een jonge man".

Nu is het vrij ingewikkeld om de Gaussiaanse klokfuctie van het voorbeeld op een simpele manier om te zetten in een wiskundige uitdrukking. Vandaar dat in de meeste gevallen de opgestelde lidmaatschapsfuncties worden vereenvoudigd. De mate van vereenvoudiging wordt de *tolerantie* genoemd. De lidmaatschapsfunctie van de linguïstische variabele "een jonge man" zou bijvoorbeeld vereenvoudigd kunnen worden tot figuur 7/7.1-3, een driehoeksvorm die gemakkelijk in een wiskundige formule omgezet kan worden.



Figuur 7/7.1-3: De lidmaatschapsfunctie van de linguïstische variabele "een jonge man" wordt vereenvoudigd tot een driehoekvorm.

Soorten lidmaatschapsfuncties

In de meeste gevallen zal men de lidmaatschapsfuncties vereenvoudigen tot:

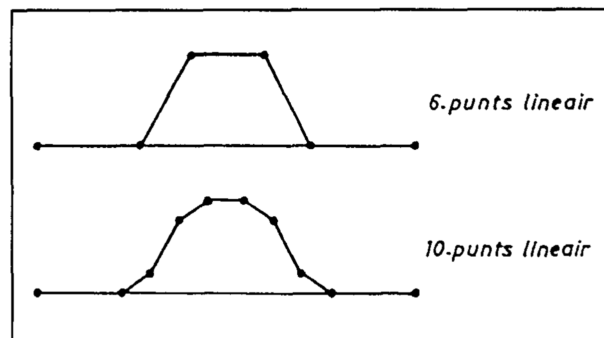
- lineaire functies;
- gesegmenteerde lineaire functies;
- S-, PI- en Z-functies;
- Singletons.

Lineaire functies

Het beschreven proces noemt men de *lineaire fuzzificering* en de daaruit volgende lidmaatschapsfuncties *lineaire functies*. Nogal logisch, want de lidmaatschapsfunctie bestaat nu nog slechts uit rechte lijntjes.

Gesegmenteerde lineaire functies

De tolerantie in bij lineaire functies vrij groot. Men kan het ook iets nauwkeuriger doen, door de Gaussiaanse klokfunctie te benaderen door middel van drie of zelfs zeven rechte lijnstukken, die de curve zo goed mogelijk benaderen. Dan ontstaan de LF's die voorgesteld worden in figuur 7/7.1-4.



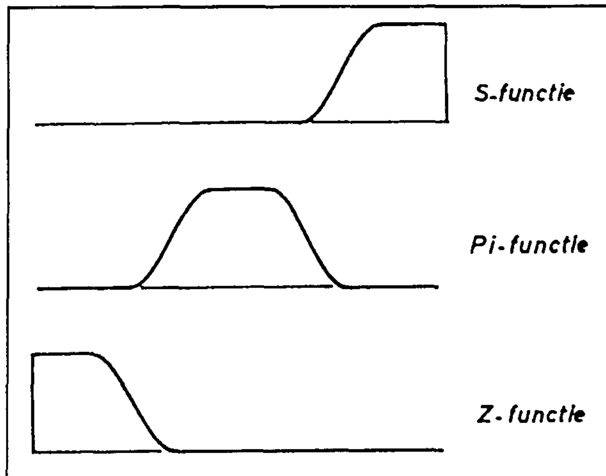
Figuur 7/7.1-4: De tolerantie neemt af als men werkt met gesegmenteerde lineaire lidmaatschapsfuncties.

Dergelijke benaderingen noemt men *gesegmenteerde lineaire lidmaatschapsfuncties*. De oorsprong van deze benaming is duidelijk: de functie bestaat nu immers uit een aantal rechte segmenten.

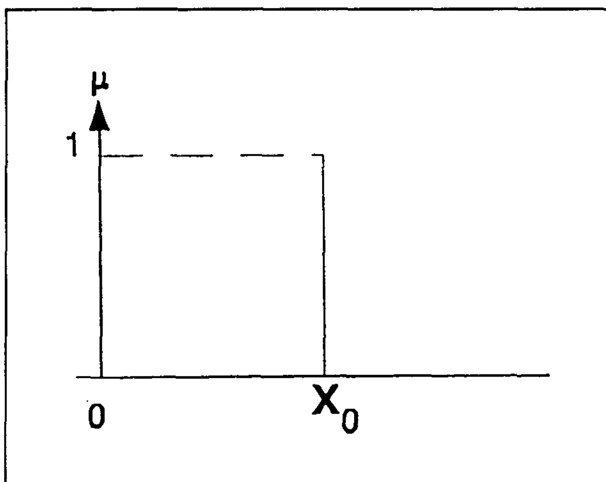
S-, PI- en Z-functies

Naast de al dan niet gesegmenteerde lineaire lidmaatschapsfuncties worden in de praktijk ook zogenoemde kwadratische functies toegepast. Er zijn drie standaardvormen, getekend in figuur 7/7.1-5. De S-, PI- en Z-functie kunnen alle drie met dezelfde vrij

eenvoudige wiskundige kwadratische vergelijkingen beschreven worden.



Figuur 7/7.1-5: De drie standaardvormen van de kwadratische lidmaatschapsfunctie.



Figuur 7/7.1-6: De singleton is een lidmaatschapsfuctie, waarvan μ_x slechts twee waarden heeft: 0 of 1.

De singleton

Een speciale lineaire lidmaatschapsfunctie is de *singleton*. Een singleton, voorgesteld in figuur 7/7.1-6, heeft slechts één lid van de vage verzameling, waarvoor μ een waarde heeft die niet nul is. Uit de aard der zaak moet deze waarde dan gelijk zijn aan 1. In feite is er aan een singleton niets vaags te ontdekken, en zou een dergelijke verzameling zich heel goed thuis voelen in de harde logica. Het begrip singleton is in de vage logica ingevoerd, omdat men nu eenmaal soms te maken heeft met variabelen die maar één waarde kunnen hebben. Een typisch voorbeeld van een linguïstische variabele, die alleen maar door middel van een singletonfunctie is uit te drukken is "noem de dag van de week die overeen komt met de datum 5 januari 1996". Hierop is maar één antwoord mogelijk, namelijk "vrijdag". Als figuur 7/7.1-6 de lidmaatschapsfunctie van deze linguïstische variabele zou voorstellen, dan zou X_0 gelijk zijn aan het antwoord "vrijdag". Uit de aard der zaak is lidmaatschapsgraad μ_{X0} gelijk aan 1.

Verfijnen van de lidmaatschapsfunctie

De vage logica heeft een aantal begrippen ingevoerd, waarmee het mogelijk wordt lidmaatschapsfuncties beter te definiëren. De voornaamste verfijningen zijn:
- termen;
- hedges.

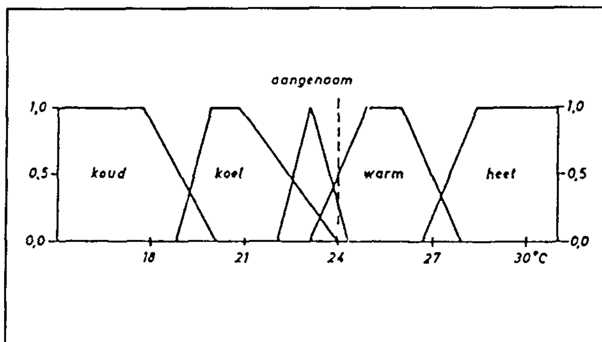
Termen

Termen zijn bijvoeglijke naamwoorden, die aan de linguïstische variabele gekoppeld worden. Ook dit begrip kan het best aan de hand van een voorbeeld uitgediept worden. Stel de linguïstische variabele "kamertemperatuur". Dat is een zeer vaag begrip, dat echter beter gedefinieerd kan worden door er enige bijvoeglijke naamwoorden of termen aan te koppelen. Men zou de kamertemperatuur kunnen omschrijven als:
- koud;
- koel;
- aangenaam;
- warm;
- heet.

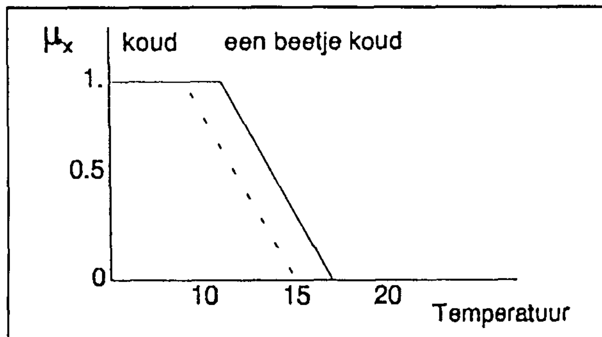
Als men nu aan duizend personen een lijstje zou overhandigen met daarin ingevuld temperaturen van 10 °C tot 32 °C en zou vragen

7.1 Achtergrond-informatie

aan iedere temperatuur een van de bovenstaande vijf termen te koppelen, dan zouden vijf vage verzamelingen ontstaan, die allemaal op de beschreven manier omgezet kunnen worden in een lidmaatschapsfunctie. Men kan deze vijf LF's verzamelen in één grafiek, met als gevolg dat figuur 7/7.1-7 ontstaat.



Figuur 7/7.1-7: De lidmaatschapsfuncties van vijf termen van de linguïstische variabele "kamertemperatuur".



Figuur 7/7.1-8: De werking van een shifted hedge op de lidmaatschapsfunctie van een term.

Merk op dat bepaalde temperaturen nu deel uit maken van verschillende verzamelingen. Dat is een zeer fundamenteel verschil tussen de harde logica en de vage logica! De temperatuur "24 °C" maakt bijvoorbeeld deel uit van de verzameling "aangenaam", maar ook van de verzameling "warm". Aan deze temperatuur kan men dus een $\mu_{aangenaam}$ en een μ_{warm} koppelen. In de traditionele harde logica is het absoluut ondenkbaar dat een element deel uitmaakt van twee verzamelingen.

De kracht van de vage logica

Het feit dat een element deel kan uitmaken van meer dan één vage verzameling is de kracht van de vage logica. Door linguïstische variabelen om te zetten in lidmaatschapsfuncties van een aantal termen, ontstaat een zeer verfijnd beeld van de variabele, een beeld dat als het ware op een menselijke manier met de variabele omgaat. Hierdoor kunnen regelsystemen als het ware natuurlijk afgestemd worden op het menselijk gedrag.

Hedges

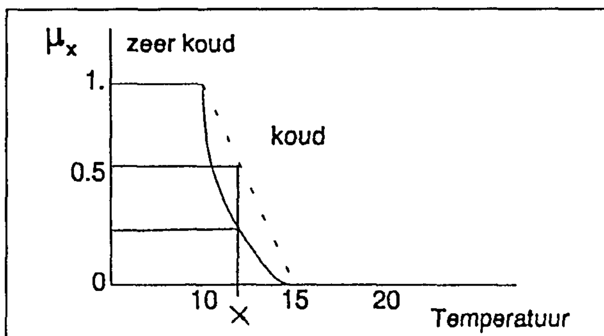
In taalkundige termen uitgedrukt zou men hedges kunnen opvatten als bijwoorden, die aan de termen van de linguïstische variabele worden gekoppeld. De term "koel" van de linguïstische variabele "kamertemperatuur" zou bijvoorbeeld nader gespecificeerd kunnen worden met de hedges "een beetje" en "zeer".

Er ontstaan dan twee tamelijk nauwkeurige linguïstische begrippen, namelijk "een beetje koele kamertemperatuur" en "zeer koele kamertemperatuur". Dank zij deze hedges kan de ontwerpen van een vaag regelsysteem ingrijpen in de lidmaatschapsfuncties. Het zal immers duidelijk zijn dat het invoeren van hedges invloed heeft op de vorm van de lidmaatschapsfuncties. Hedges worden vaak ingevoerd in de testfase van een vaag regelsysteem. Als dan blijkt dat de opgestelde lidmaatschapsfuncties van de in- en uitgangsgrootheden toch niet het gewenste regeleffect veroorzaken, dan kan men het systeem verfijnen door de lidmaatschapsfuncties iets aan te passen door het invoeren van een of meerdere hedges.

Hiervoor zijn twee benaderingen mogelijk:
- shifted hedges;
- powered hedges.

Shifted hedges
Een shifted hedge doet niets anders dan de grenzen van de lidmaatschapsfunctie van de term naar links of naar rechts verschuiven. In figuur 7/7.1-8 is aangegeven hoe de hedge "een beetje" de LF van de term "koud" uit figuur 7/7.1-7 naar rechts verschuift.

Powered hedge
Bij de powered hedge worden de grenzen van de lidmaatschapsfunctie niet verschoven, maar wordt de vorm ervan aangepast. In figuur 7/7.1-9 is getekend hoe door het invoeren van de powered hegde "zeer" in de term "koud" de lidmaatschapsfunctie zo wordt aangepast dat lagere temperaturen in de verzameling "koud" een minder dan lineaire toename van hun μ krijgen.



Figuur 7/7.1-9: De invloed van een powered hedge op de lidmaatschapsfunctie van een term.

Een bepaalde temperatuur X heeft zonder de hedge een μ_X van ongeveer 0,5. Door het invoeren van de powered hedge daalt zijn lidmaatschapsgraad μ tot ongeveer 0,25.

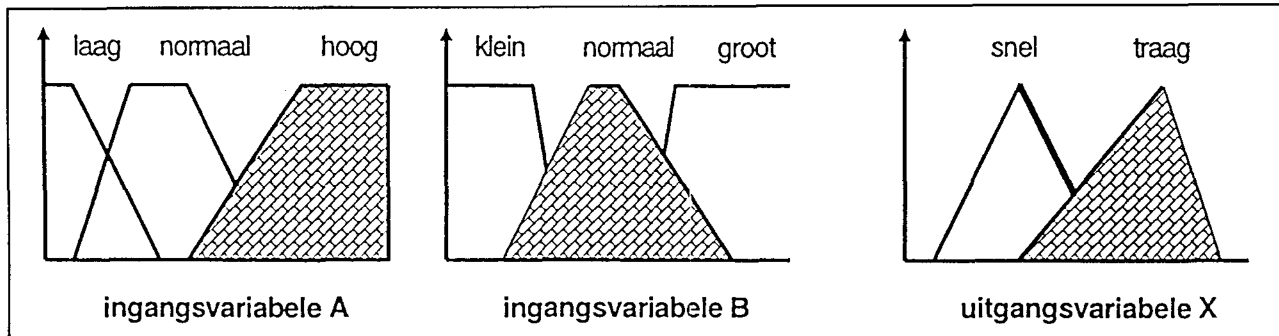
Het redeneermechanisme

Inleiding
Doel van het redeneermechanisme is een aantal regels opstellen, die de gewenste relatie tussen de ingangs- en de uitgangsgrootheden vast leggen. In principe wijkt deze stap niet af van het opstellen van de beslissingsregels in de harde logica. Het enig verschil is dat de beslissingsregels in de vage logica minder hard zijn en meer aangepast aan de menselijke subjectieve maatstaven.
De meest algemene vorm van een vage beslissingsregel is:
ALS *premisse* **DAN** *conclusie*

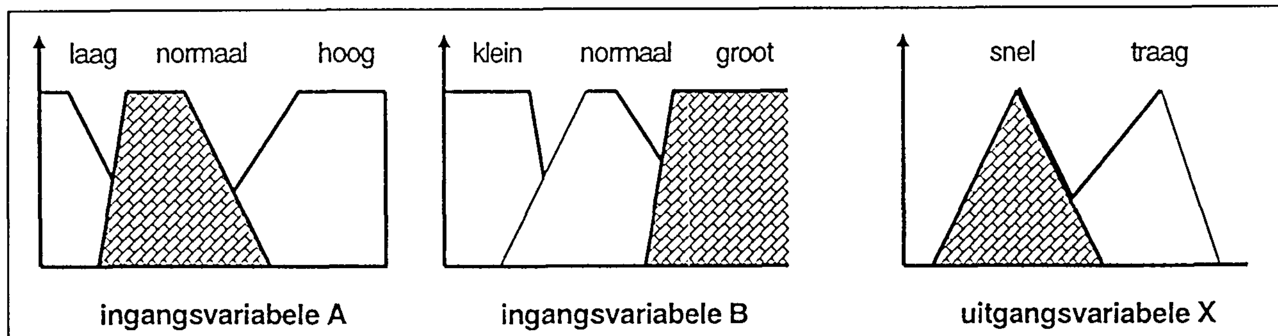
Premisse
De premisse bestaat uit een of meer uitspraken, *antecendenten* genoemd, die betrekking hebben op de ingangsgrootheden. Een voorbeeld van een premisse is:
"kamertemperatuur IS warm EN buitentemperatuur IS koel".
Twee linguïstische ingangsvariabelen "kamertemperatuur" en "buitentemperatuur" worden hierin nader omschreven door hun termen "warm" en "koel" en door middel van een fuzzy-operator EN aan elkaar gekoppeld. De premisse geeft dus als het ware aan welke lidmaatschapsfuncties van de ingangsgrootheden in de regel actief zijn. Bij het uitwerken van deze regel zal de fuzzyprocessor de actuele waarden van de twee ingangsgrootheden meten, deze op de assenstelsels van de LF's zetten en de μ's berekenen van de actieve lidmaatschapsfuncties. Deze twee waarden van μ worden dan gebruikt om de uitgangsvariabele te berekenen.

Conclusie
De *conclusie* heeft op dezelfde manier betrekking op de uitgangsgrootheden. In het voorbeeld:
motorsnelheid IS hoog
wordt de LF van term "hoog" van de linguïstische uitgangsvariabele "motorsnelheid" geactiveerd. De berekende μ's van de actieve ingangs-LF's worden aan de geselecteerde uitgangs-LF aangeboden en er wordt op een bepaalde manier een specifieke waarde voor de uitgangsvariabele berekend.
Op dit rekenmechanisme wordt later nader ingegaan.

7.1 Achtergrond-informatie



Figuur 7/7.1-10: Grafische toelichting van de reële betekenis van de premisse en de conclusie.



Figuur 7/7.1-11: De actieve LF's voor de vage regel "Als A is normaal EN B is groot, DAN X is snel".

Voorbeeld van premisse en conclusie

Aan de hand van een voorbeeldje wordt de praktische betekenis van premisse en conclusie toegelicht. Stel dat een bepaald systeem een uitgangsvariabele X heeft, die op een bepaalde manier moet reageren op de waarde van twee ingangsvariabelen A en B. Volgens de regels van de fuzzy kunst worden deze drie linguïstische grootheden nader gespecificeerd door termen. A kan ingedeeld worden in de drie termen "laag", "normaal" en "hoog". Variabele B krijgt ook drie termen, namelijk "klein", "normaal" en "groot". De uitgangsvariabele X wordt met de termen "snel" en "traag" beschreven. Vervolgens stelt men voor de acht termen lidmaatschapsfuncties op en tekent deze in drie grafieken. Een en ander is toegelicht in figuur 7/7.1-10.

Vervolgens stelt men een vage beslissingsregel op:

ALS A IS hoog EN B IS normaal DAN X IS traag

Het fuzzy-systeem weet nu dat bij de uitwerking van deze regel het rekening moet houden met de LF van "A-hoog" en de LF van "B-normaal". De overige LF's van de ingangsvariabelen zijn niet aan de orde. Het resultaat zal uitgelezen moeten worden in de LF van "X-traag". Hoe dat in- en uitlezen gaat wordt later besproken.

Men kan nu voor dit systeem een tweede vage beslissingsregel opstellen, bijvoorbeeld:

ALS A IS normaal EN B IS groot, DAN X is snel

In figuur 7/7.1-11 is getekend welke lidmaatschapsfuncties nu actief worden.

Verfijningen

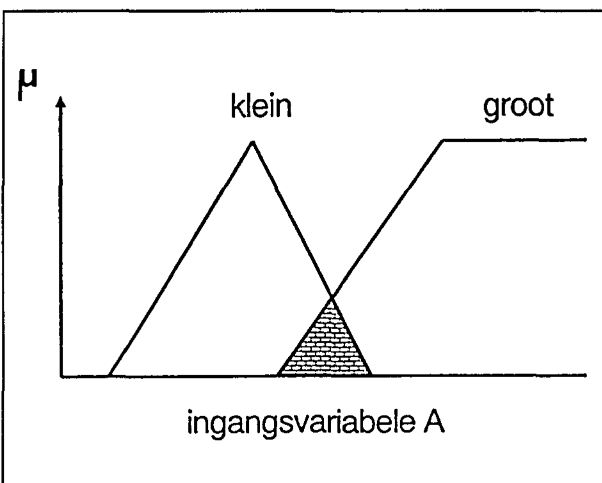
Uiteraard stelt de theorie een aantal verfijningen in, waardoor een beslissingsregel veel nauwkeuriger het verband tussen in- en uitgangsgrootheden kan vastleggen. De voornaamste verfijningen zijn:

- fuzzy-operatoren;
- evaluerende regels;
- voorspellende regels.

Fuzzy-operatoren
Net zoals de harde Booleaanse algebra kent ook het rekenstelsel van de fuzzy logic een aantal operatoren, te weten:
- EN;
- OF;
- NIET.

Men kan deze operatoren opnemen in een premisse om de onderlinge relatie tussen de ingangsgrootheden beter te omschrijven.

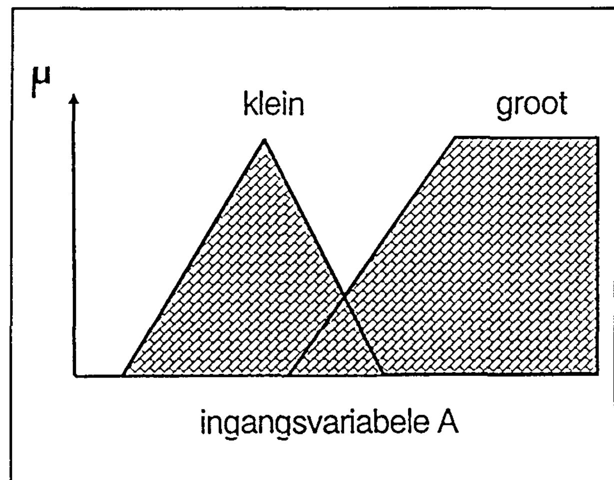
De EN-operator
De EN-operator laat toe twee lidmaatschapsfuncties van een variabele actief te maken in één beslissingsregel. Zoals uit figuur 7/7.1-12 blijkt, zal bij een EN-koppeling alleen het gemeenschappelijk oppervlak van beide LF's actief worden. De figuur geeft het resultaat van:
ALS A IS klein EN A is groot DAN ....



Figuur 7/7.1-12: Het resultaat van een EN-operator op twee lidmaatschapsfuncties van een variabele.

Natuurlijk wordt de EN-operator ook gebruikt om twee ingangsvariabelen met elkaar te koppelen, zoals in:
Als A is groot EN B IS klein DAN ....

In dit geval zal de kleinste μ-waarde van beide LF's in de berekening betrokken worden. Dit wordt toegelicht aan de hand van figuur 7/7.1-13 (blz. 11). De vage beslissingsregel van dit voorbeeld luidt:
ALS A IS normaal EN B IS groot DAN ...
Op het moment dat de fuzzy-processor deze regel verwerkt, meet hij de actuele waarden van de ingangsvariabelen. Deze zijn A1 en B1. Deze worden op de assen van de functies gezet.
Hieruit kan men twee actieve μ-waarden berekenen, een voor A1 (μ_{A1}) en een voor B1 (μ_{B1}). Omdat A en B in de regel door een EN-operator gekoppeld zijn, zal nu alleen de laagste μ-waarde overgedragen worden naar de uitgangsfunctie voor het berekenen van de actuele waarde van de uitgang. In dit voorbeeld is dit dus duidelijk μ_{A1}.

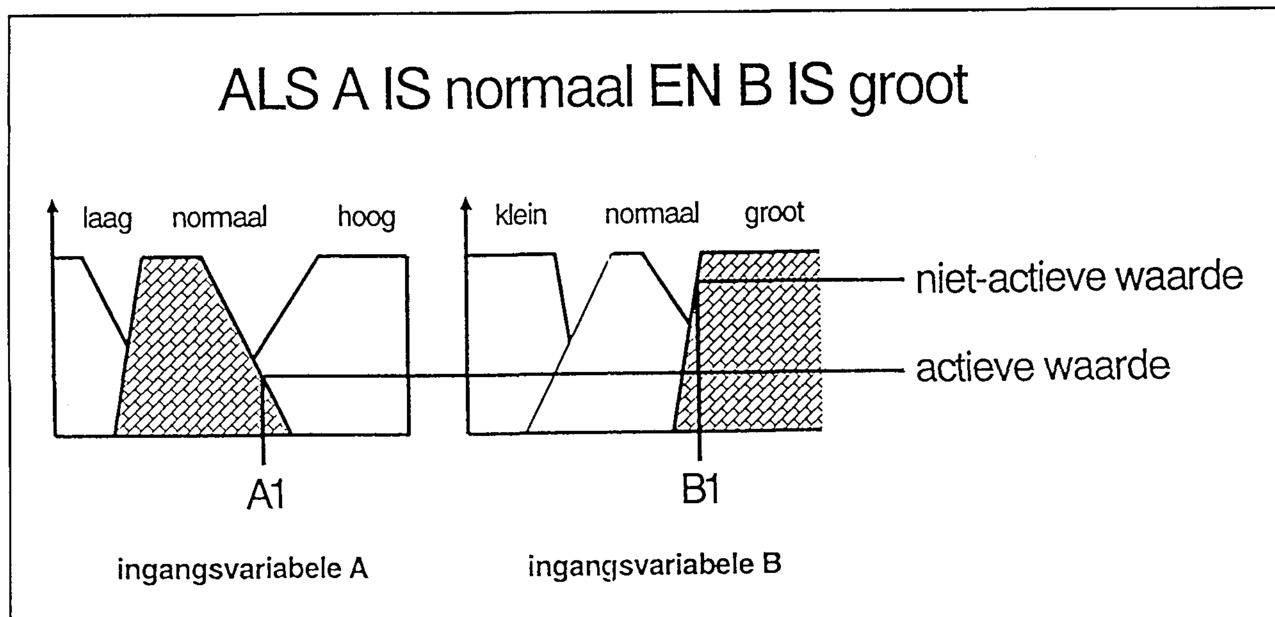


Figuur 7/7.1-14: Het koppelen van twee LF's door middel van een OF-operator.

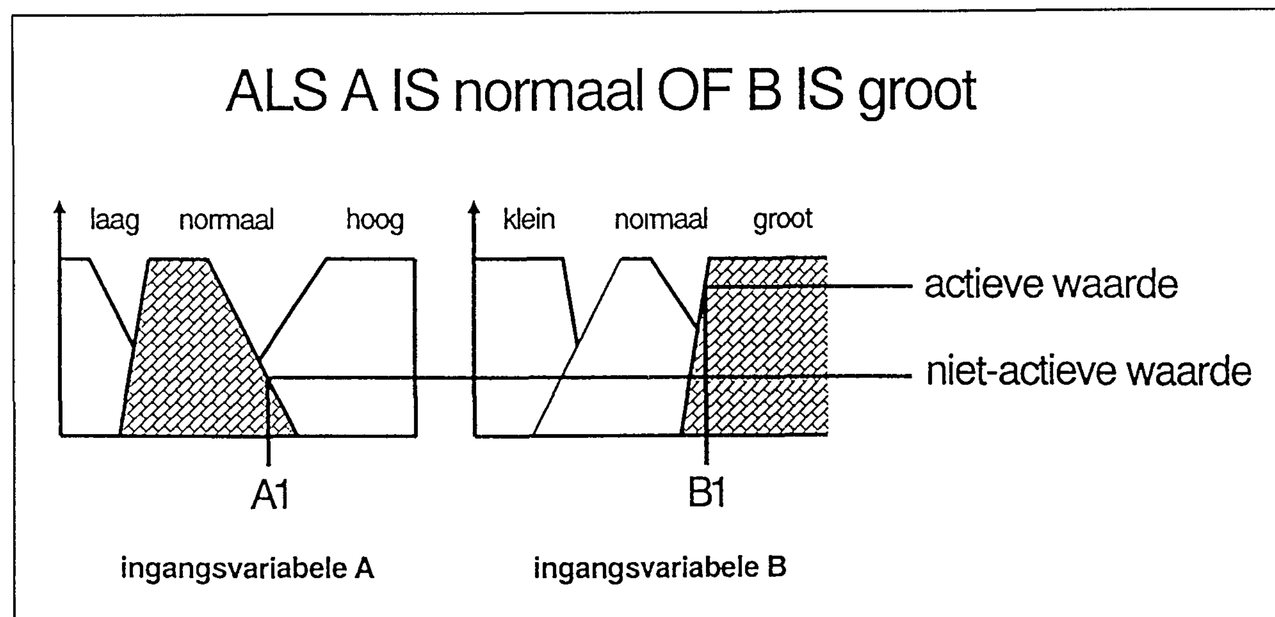
De OF-operator
Ook deze operator laat toe twee LF's van een ingangsvariabele te activeren. In dit geval wordt, zoal getekend in figuur 7/7.1-14, het gezamenlijk oppervlak van beide LF's geactiveerd. De premisse van dit voorbeeld luidt:
ALS A IS klein OF A IS groot DAN ....

7.1 Achtergrond-informatie

Natuurlijk kan men de OF-operator ook gebruiken om twee variabelen te koppelen. In figuur 7/7.1-15 is als voorbeeld de premisse:
ALS A IS normaal OF B IS groot DAN . . .
Ook nu meet de fuzzy-processor de actuele waarden van de ingangen en berekent uit de actieve LF's hun μ-waarden. Maar vanwege de OF-koppeling wordt nu de grootste μ-waarde doorgekoppeld naar de uitgangsfunctie.
In dit geval is dit dus μ_{B1}.



Figuur 7/7.1-13: De invloed van een EN-operator op het evalueren van de waarden van de premisse.



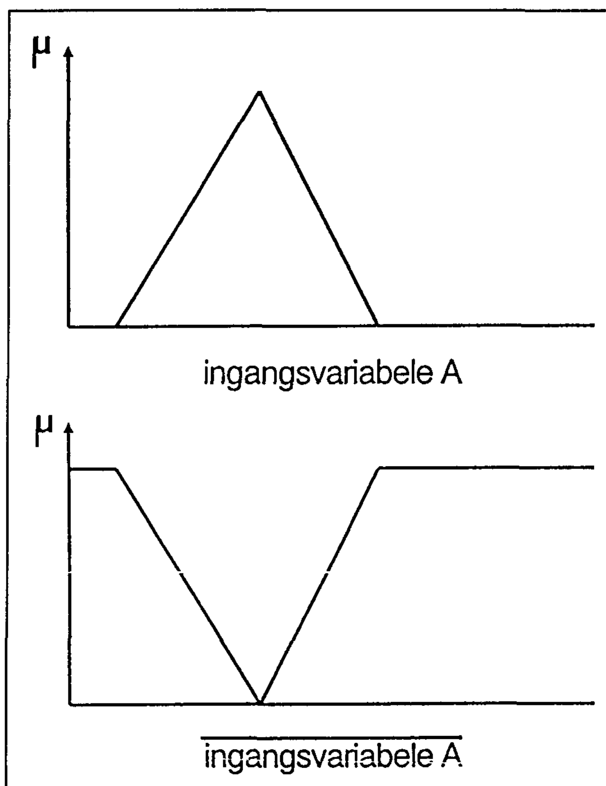
Figuur 7/7.1-15: Het koppelen van twee ingangsvariabelen door middel van een OF-operator.

De NIET-operator

De NIET-operator is de ook in de harde logica bekende inverteer-functie. Het resultaat van een NIET-operatie is dat alle waarden van μ opnieuw worden berekend volgens de formule:

$\mu\bar{A}(x) = 1 - \mu A(x)$

Met andere woorden: de geïnverteerde waarden $\bar{\mu}$ van μ kan men berekenen door de waarden van μ van 1 af te trekken. Wat dit grafisch betekent is voorgesteld in figuur 7/7.1-16.



Figuur 7/7.1-16: Het resultaat van een NIET-bewerking op een lidmaatschapsfunctie.

Evaluerende regels

Evaluerende regels worden het vaakst toegepast. Alle tot nu toe gebruikte voorbeelden zijn evaluerende regels. De meest algemene uitdrukking van een evaluerende regel is:

ALS A IS a EN/OF B IS b DAN X is x

waarin staan A en B voor ingangsvariabelen, a en b voor hun actieve termen, X voor de uitgangsvariabele en x voor zijn actieve term.
De meeste opdrachten uit de vage logica kunnen uitstekend met evaluerende regels opgelost worden.

Voorspellende regels

Voorspellende regels zijn niet zo gemakkelijk op te stellen, maar zij bieden wel zeer krachtige functies.
Een voorbeeld van een voorspellende regel, zuiver onder linguïstische vorm is:
"ALS krachtiger remmen bij een gegeven snelheid tot gevolg heeft dat de motor sneller stopt binnen de toegestane afstand, ga DAN krachtige remmen"
Een specifieke eigenschap van voorspellende regels is dat de uitgangsvariabele altijd wordt opgenomen binnen de premisse van de regel. Er kunnen een aantal voorspellende regels onder elkaar in het programma worden opgenomen. De fuzzy-processor zal deze dan een voor een evalueren en de voorspellende regel die het beste stuursignaal oplevert vervolgens selecteren en uitvoeren.
In termen van fuzzy notatie kan een voorspellende regel als volgt geschreven worden:

ALS [X IS x ⇒ (A IS a EN/OF B IS b)] DAN X IS x

De inferentie

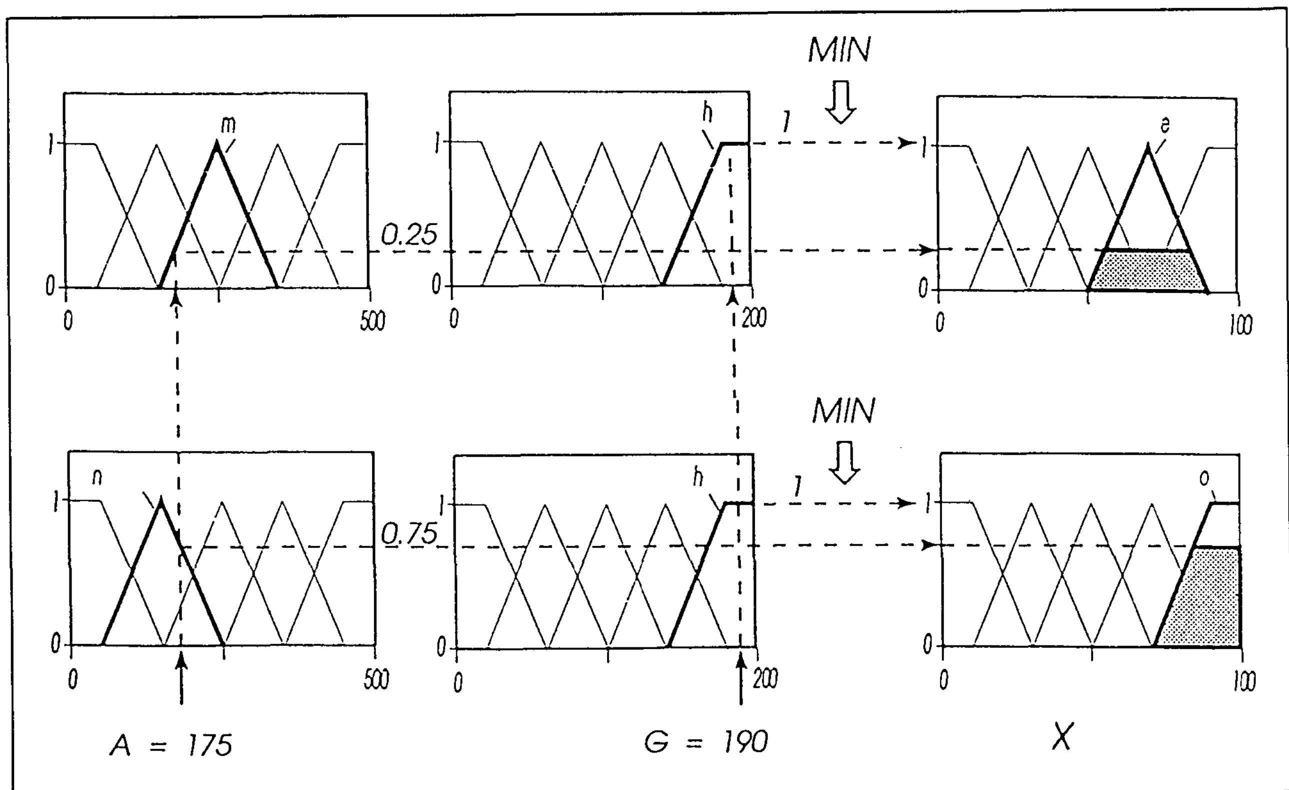
Inleiding

De *inferentie* is de techniek van het redeneer mechanisme, waarmee de fuzzy-processor uit de opgestelde regels een bepaalde uitgangsfunctie afleidt. Deze uitgangsfunctie hangt bovendien af van de momentele waarden van de μ's van de ingangen, maar is nog steeds een vage verzameling die men een fuzzy-functie noemt.
Een voorbeeld zal het inferentie-proces verduidelijken.

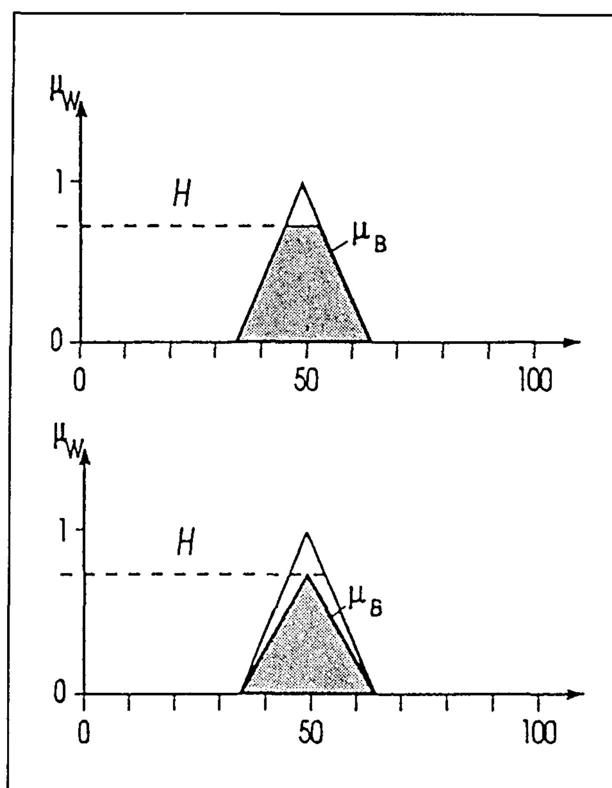
Voorbeeld

- Gegeven:
  een regelsysteem met twee ingangen A en G en een uitgang X
- Fuzzicatie:
  - ingang A kan beschreven worden in vijf termen, waaronder m en n;
  - ingang G kan beschreven worden in vijf termen, waaronder h;
  - uitgang X kan beschreven worden in vijf termen, waaronder e en o.
- Redeneermechanisme:
  Het te sturen proces kan volledig beschreven worden met twee vage regels:
  - Regel 1:
    ALS A IS m EN G IS h DAN X is e
  - Regel 2:
    ALS A IS n EN G IS h DAN X IS o
- Actuele ingangssituatie:
  Op het moment dat de fuzzy-processor de regels uitwerkt is de actuele waarde van de ingangsgrootheden:
  A = 175
  G = 190

Het volledige proces van inferentie is samengevat in figuur 7/7.1-17.
De bovenste grafiek geeft de inferentie van de eerste regel. Natuurlijk zijn de LF's m van A en h van G actief, want die staan vermeld in de eerste regel. Uit de actuele waarden 175 en 190 kunnen de twee corresponderende μ-waarden berekend worden, waaruit blijkt dat $\mu_A = 0{,}25$ en $\mu_G = 1$. Omdat beide variabelen door middel van een EN-operator gekoppeld zijn, telt alleen de laagste μ-waarde, in dit geval 0,25.
Deze wordt horizontaal doorgetrokken naar de grafiek van de uitgang X. Hier is de LF van e actief. Er ontstaat nu een nieuwe vage verzameling, die begrensd wordt door de μ-waarde 0,25. Deze fuzzy-functie is gearceerd weergegeven.



Figuur 7/7.1-17: Het principe van inferentie toegelicht aan de hand van een voorbeeld.



Figuur 7/7.1-18: Twee vormen van inferentie-strategie: boven MIN-MAX, onder MAX-PROD.

De onderste grafiek geeft de inferentie van de tweede regel. Nu is de LF A_n actief, alsmede de LF van G_h. De waarden van de ingangen zijn natuurlijk nog niet gewijzigd, met als gevolg dat er nu twee μ-waarden berekend kunnen worden van 0,75 en 1. Ook nu wordt er gekoppeld met een EN-operator, zodat de laagste μ (0,75) weer wordt doorgetrokken naar de uitgangsgrafiek. Hier is de LF X_o actief. Ook nu berekent het inferentieproces een nieuwe vage verzameling, die nu begrensd wordt door de μ-waarde 0,75.

Soorten inferentie
Er zijn verschillende algoritmen ontwikkeld voor het opstellen van fuzzy-functies door middel van inferentie. De meest toegepaste zijn:
- MIN-MAX inferentie;
- MAX-PROD inferentie.

De verschillen worden toegelicht aan de hand van figuur 7/7.1-18.

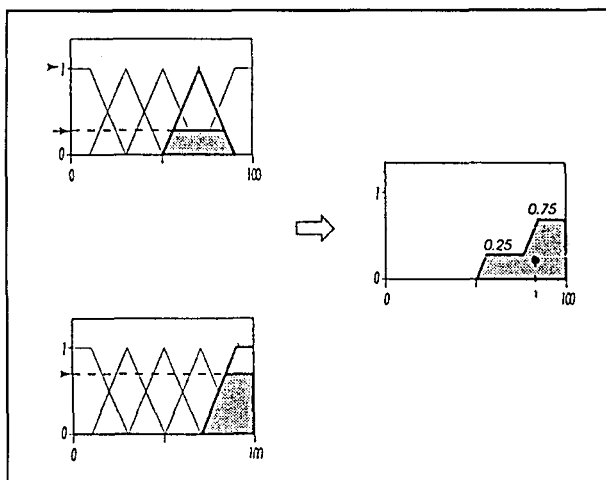
MIN-MAX inferentie
De bovenste tekening geeft het resultaat van een MIN-MAX inferentie. De actieve LF van de uitgang wordt begrensd door de actieve waarde van de μ van de ingangen. Hieruit volgt dat het inferentie-voorbeeld van figuur 7/7.1-17 volgens het MIN-MAX algoritme werkt.

MAX-PROD inferentie
De onderste tekening toont wat er gebeurt als wordt gewerkt volgens het MAX-PROD schema. Nu wordt de actieve LF van de uitgang niet begrensd, maar verkleint in hoogte, tot de top samenvalt met de actieve waarde van μ.

Inferentie-strategieën
Er is geen sprake van een "goede" inferentie-strategie. Voor sommige toepassingen kan men het best gebruik maken van de MIN-MAX algoritmen, andere toepassingen geven de beste resultaten als men volgens MAX-PROD werkt. Bovendien zijn er tientallen andere inferentie-strategieën theoretisch uitgewerkt, die weliswaar op dit moment nog niet vaak worden toegepast in reële toepassingen van fuzzy techniek, maar toekomstperspectieven bieden.

De compositie
Door de inferentie wordt iedere regel van het redeneer mechanisme omgezet in een fuzzy-functie, die het door de uitgang te volgen gedrag definieert bij bepaalde waarden van de ingangsvariabelen. Natuurlijk heeft men daar in de praktijk vrij weinig aan, want wat moet ontstaan is een eenduidige uitgangsfunctie, waaruit een welbepaalde waarde voor de uitgang kan worden afgeleid. Op de een of andere manier moeten alle afzonderlijke fuzzy-functies van de uitgang worden gecombineerd tot één algemeen geldende functie, die rekening houdt met alle stellingen

die in alle regels vast gelegd zijn. Dit proces noemt men de *compositie*.
In de meeste gevallen gaat dit door de afzonderlijke fuzzy-functies met elkaar te verknopen met een OR-operator. Dat is vrij logisch, want men kan er van uitgaan dat het systeem aan alle regels van het redeneer mechanisme moet voldoen, hetgeen te vertalen is naar een OR-functie.
In het voorbeeld van figuur 7/7.1-17 levert de compositie dus een eenduidige uitgangsfunctie op, die ontstaat door de twee deelfuncties te OR-ren. Hetgeen, zoals reeds beschreven, er op neer komt dat de oppervlakken van de deelfuncties tot één oppervlak worden verenigd. Een en ander wordt grafisch toegelicht in figuur 7/7.1-19.



Figuur 7/7.1-19: Door een OR-operator toe te passen op de twee deel-functies ontstaat de uiteindelijke uitgangs-functie van het fuzzy-proces.

De defuzzificatie

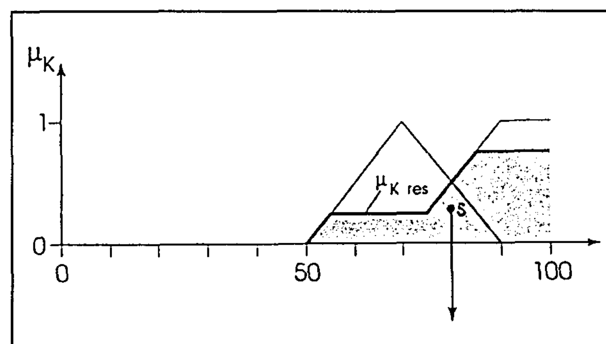
Inleiding

De uitgangs-functie die door de compositie ontstaat bepaalt volledig de reactie van de uitgang op de momentele ingangswaarden. Natuurlijk moet uit deze functie een expliciete waarde van de uitgangsvariabele worden afgeleid. Dit proces noemt men de *defuzzificatie*.
Ook hiervoor zijn verschillende strategieën ontwikkeld, waarvan de voornaamste zijn:
- de center of gravity strategie COG;
- de maximum strategie MAX;
- de mean of maximum strategie MOM.

De center of gravity strategie wordt het vaakst toegepast.

De center of gravity strategie

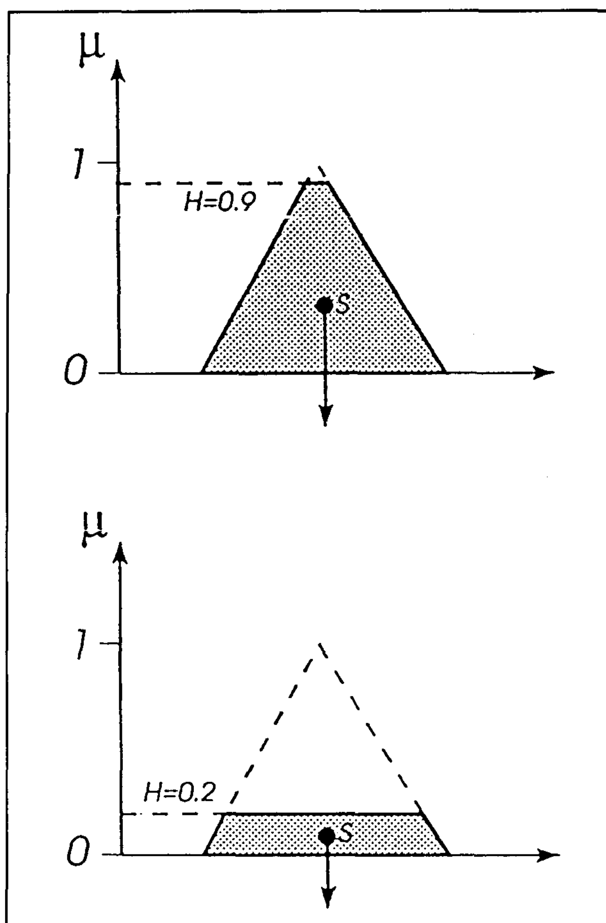
Bij deze strategie bepaalt de fuzzy-processor het zwaartepunt S van de uitgangs-functie. Het zwaartepunt of *centroide* is een wiskundig begrip, dat soms met eenvoudige en soms met ingewikkelde wiskunde op ieder plat vlak kan worden toegepast. In figuur 7/7.1-20 is getekend waar het zwaartepunt van de uitgangsfunctie van figuur 7/7.1-19 ligt. Door uit dit zwaartepunt S een vertikaal hulplijntje te tekenen naar de horizontale as kan de numerieke waarde van de uitgangsvariabele afgelezen worden.
In het voorbeeld betekent dit dat het fuzzy systeem de uitgangsvariabele X een waarde van 80 geeft, als de ingangsvariabele A gelijk is aan 175 en de ingangsvariabele G gelijk is aan 190.



Figuur 7/7.1-20: Het bepalen van de numerieke waarde van de uitgangsvariabele door middel van de COG-strategie.

De COG-strategie heeft een groot nadeel, dat toegelicht wordt aan de hand van figuur 7/7.1-21. In die tekening zijn twee uitgangs-

functies getekend, die duidelijk een andere vorm hebben. Als men van beide functies het zwaartepunt berekent en daarvan de horizontale waarde, dan blijkt dat deze waarden gelijk zijn! In beide gevallen zou de uitgangsvariabele dus dezelfde numerieke waarde krijgen, terwijl het duidelijk is dat de ingangsvariabelen niet gelijk zijn! Ondanks dit bezwaar schijnt de COG-strategie in de praktijk zeer goede resultaten op te leveren.

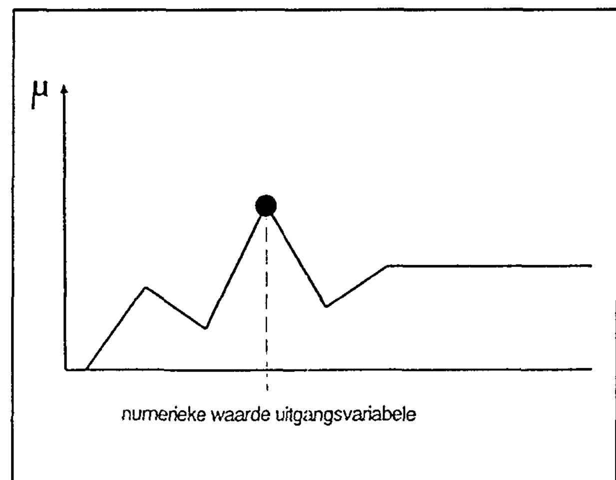


Figuur 7/7.1-21: Een nadeel van de COG-strategie is dat twee verschillende uitgangs-functies dezelfde numerieke waarde voor de uitgang kunnen opleveren.

Een tweede nadeel van de COG-strategie is dat het bepalen van het zwaartepunt bij ingewikkelde, niet-lineaire uitgangs-functie een hele klus is, waar zelfs een snelle processor de nodige tijd over doet.

De maximum strategie

Bij de MAX-strategie wordt het punt van de uitgangs-functie bepaald dat de hoogste waarde op de μ-as heeft. Vanuit dit punt wordt weer een vertikale lijn naar de horizontale as getrokken, waar de numerieke waarde van de uitgangsvariabele afgelezen kan worden. In figuur 7/7.1-22 is dit toegelicht aan de hand van een voorbeeldje.
Voordeel van de MAX-strategie is dat het bepalen van het maximum van een curve wiskundig heel snel gaat en dat de processor dus weinig rekentijd nodig heeft voor de defuzzificatie.



Figuur 7/7.1-22: Defuzzificatie volgens de MAX-strategie.

De mean of maximum strategie

De MOM-defuzzificatie werkt volgens een heel eigen principe. Er wordt namelijk géén compositie toegepast. Iedere regel uit het redeneer mechanisme wekt, zoals bekend, een specifieke functie op voor de uitgang. Bij de MOM-strategie wordt van iedere uitgangsfunctie het maximum bepaald en de bijbehorende μ_{max} berekend. Nadien wordt het gemiddelde van al deze waarde berekend en dit getal wordt gebruikt om de reële waarde van de uitgangsvariabele te bepalen.

Het voordeel van deze strategie is dat de fuzzy-processor veel minder hoeft te rekenen en daardoor veel sneller kan werken.

Conclusie

Samenvatting
Hiermee is het gehele fuzzy-proces stap na stap besproken. Een samenvatting lijkt op zijn plaats. Het oplossen van een regelprobleem door middel van vage logica gaat als volgt:

- Stap 1:
  Bepaal wat de in- en de uitgangsvariabelen zijn.
- Stap 2:
  Bepaal de minimum en maximum grenzen van deze variabelen.
- Stap 3:
  Onderzoek hoeveel termen er nodig zijn om het gedrag van de variabelen te beschrijven.
- Stap 4:
  Stel de lidmaatschapsfuncties voor alle termen op.
- Stap 5:
  Onderzoek hoe de uitgangsvariabele zich moet gedragen in functie van de ingangsvariabelen.
- Stap 6:
  Stel aan de hand hiervan de regels van het redeneer mechanisme op.
- Stap 7:
  Bepaal welke inferentie het best kan worden toegepast.
- Stap 8:
  Bepaal welke defuzzificatie strategie het best kan worden toegepast.

7.1 Achtergrond-informatie